Power Delivery Flash MCU

HT45F9160

Revision: V1.10 Date: January 03, 2024

www.holtek.com

Table of Contents

Features

CPU Features

- Operating Voltage
  - f_{SYS}=3MHz: 2.2V~5.5V
  - f_{SYS}=12MHz: 2.7V~5.5V
  - f_{SYS}=16MHz: 3.3V~5.5V
  - f_{SYS}=20MHz: 4.0V~5.5V
- Up to 0.2μs instruction cycle with 20MHz system clock at V_{DD}=5V
- Power down and wake-up functions to reduce power consumption
- Oscillator Types
  - Internal High Speed 12/16/20MHz RC – HIRC
  - Internal Low Speed 32kHz RC – LIRC
- Fully integrated internal oscillators require no external components
- Multi-mode operation: FAST, SLOW, IDLE and SLEEP
- All instructions executed in one to three instruction cycles
- Table read instructions
- 115 powerful instructions
- 16-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Flash Program Memory: 16K×16
- Data Memory: 2048×8
- True EEPROM Memory: 1024×8
- In Application Programming function – IAP
- Internal On-Chip Debug Support function – OCDS
- Watchdog Timer function
- 19 bidirectional I/O lines
- Programmable I/O source current for LED applications
- Four external interrupt lines shared with I/O pins
- Multiple Timer Modules for time measure, input capture, compare match output, PWM output function or single pulse output function
  - Single Standard type 16-bit Timer Module – STM
  - Dual Periodic type 10-bit Timer Modules – PTM0~PTM1
  - Dual Periodic type 16-bit Timer Modules – PTM2~PTM3
- Operational Amplifier function
- USB charge/discharge auto detection function
- I^2C Interface (Master Mode)
- Serial Interface Module – SIM includes SPI and I^2C interfaces
- Fully-duplex / Half-duplex Universal Asynchronous Receiver and Transmitter Interface – UART

- Dual Time-Base functions for generation of fixed time interrupt signals
- 11 external channel 12-bit resolution A/D converter with internal reference voltage V_{VR}
- Compliant with USB PD 3.1/PPS specification
- Supports Dual Role Port – DRP
- Supports Fast Role Swap – FRS
- Integrated VCONN Switch
- Integrated Multiplier/Divider Unit – MDU
- Integrated 16-bit Cyclic Redundancy Check function – CRC
- Low voltage reset function
- Low voltage detect function
- Package type: 32-pin QFN

General Description

The device is an A/D type Flash Memory 8-bit high performance RISC architecture microcontroller, specifically designed for PD Power Bank applications.

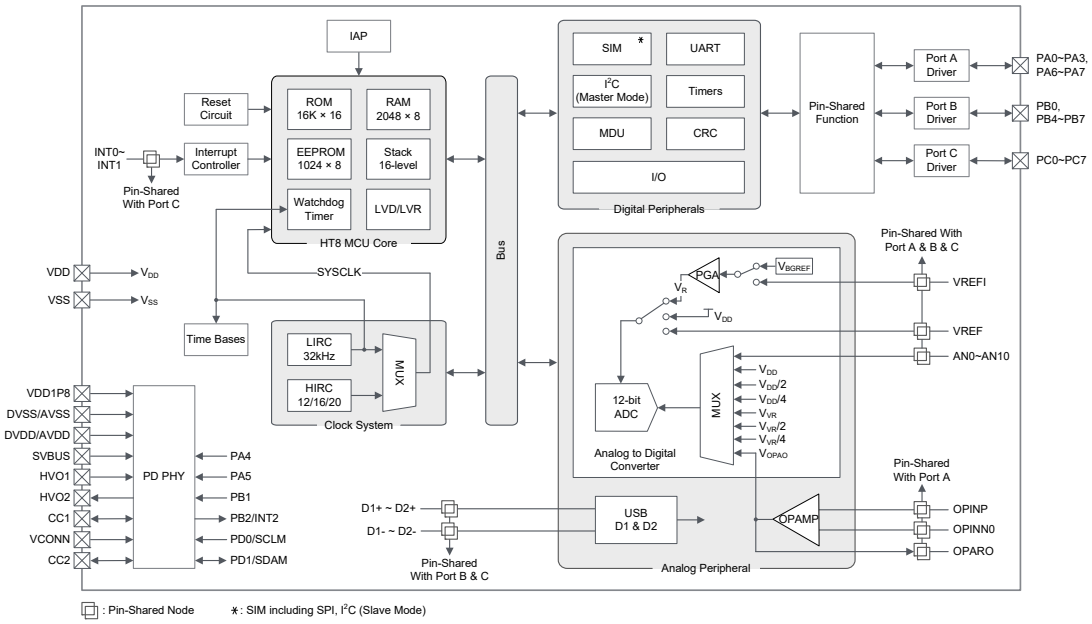
For memory features, the Flash Memory offers users the convenience of multi-programming features. Other memory includes an area of RAM Data Memory as well as an area of true EEPROM memory for storage of non-volatile data such as serial number, calibration data, etc. By using the In Application Programming technology, users have a convenient means to directly store their measured data in the Flash Program Memory as well as having the ability to easily update their application programs.

Analog features include a multi-channel 12-bit A/D converter, an USB charge/discharge auto detection function and an operational amplifier current detection function. Multiple and extremely flexible Timer Modules provide timing, pulse generation and PWM generation functions. Communication with the outside world is catered for by including fully integrated SPI, UART or I^2C (Master/Slave mode) interface functions, these popular interfaces which provide designers with a means of easy communication with external peripheral hardware. Protective features such as an internal Watchdog Timer, Low Voltage Reset and Low Voltage Detector coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

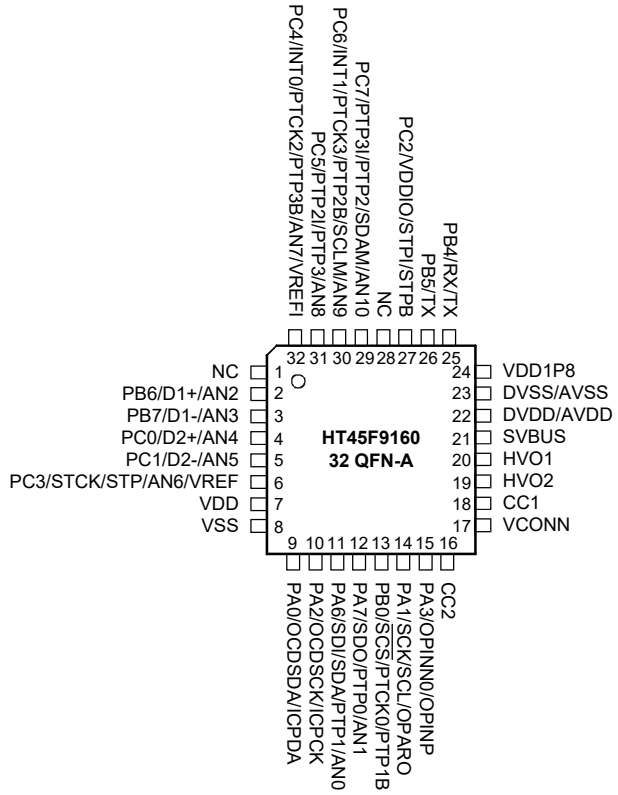
The device also includes fully integrated high and low oscillators which require no external components for their implementation. The ability to operate and switch dynamically between a range of operating modes using different clock sources gives users the ability to optimise microcontroller operation and minimise power consumption.

The device integrates USB Power Delivery (PD) PHY communication protocols and is compliant with USB PD 3.1/PPS specification. The inclusion of flexible I/O programming features, Time-Base functions, a 16-bit MDU, a CRC along with many other features ensure that the device will find excellent use in different PD Power Bank applications.

Block Diagram



Pin Assignment



Note: 1. If the pin-shared pin functions have multiple outputs simultaneously, the desired pin-shared function is determined by the corresponding software control bits.

2. The OCDSDA and OCDSCK pins are used as the OCDS dedicated pins.

Pin Descriptions

With the exception of the power pins, all pins on the device can be referenced by their Port name, e.g. PA0, PA1, etc., which refer to the digital I/O function of the pins. However these Port pins are also shared with other function such as the Analog to Digital Converter, Timer Module pins, etc. The function of each pin is listed in the following table, however the details behind how each pin is configured is contained in other sections of the datasheet.

Pin Name	Function	OPT	I/T	O/T	Description
PA0/ICPDA/OCDSDA	PA0	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	ICPDA	—	ST	CMOS	ICP Data/Address pin
	OCDSDA	—	ST	CMOS	OCDS Data/Address pin
PA1/SCK/SCL/OPARO	PA1	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	SCK	PAS0	ST	CMOS	SIM SPI serial clock
	SCL	PAS0	ST	NMOS	SIM I^2C clock line
	OPARO	PAS0	—	AN	OPAMP output
PA2/ICPCK/OCDSCK	PA2	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	ICPCK	—	ST	CMOS	ICP Clock pin
	OCDSCK	—	ST	—	OCDS Clock pin
PA3/OPINN0/OPINP	PA3	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	OPINN0	PAS0	AN	—	OPAMP negative input
	OPINP	PAS0	AN	—	OPAMP positive input
PA6/SDI/SDA/PTP1/AN0	PA6	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	SDI	PAS1	ST	—	SIM SPI data input
	SDA	PAS1	ST	NMOS	SIM I^2C data line
	PTP1	PAS1	—	CMOS	PTM1 output
	AN0	PAS1	AN	—	A/D Converter external input channel
PA7/SDO/PTP0/AN1	PA7	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	SDO	PAS1	—	CMOS	SIM SPI data output
	PTP0	PAS1	—	CMOS	PTM0 output
	AN1	PAS1	AN	—	A/D Converter external input channel
PB0/$\overline{SCS}$/PTCK0/PTP1B	PB0	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	$\overline{SCS}$	PBS0	ST	CMOS	SIM SPI slave select
	PTCK0	PBS0	ST	—	PTM0 clock input
	PTP1B	PBS0	—	CMOS	PTM1 inverted output
PB4/RX/TX	PB4	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	RX/TX	PBS1 IFS	ST	CMOS	UART serial data input in full-duplex communication or UART serial data input / output in Single Wire Mode communication
PB5/TX	PB5	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	TX	PBS1	—	CMOS	UART serial data output

Pin Name	Function	OPT	I/T	O/T	Description
PB6/D1+/AN2	PB6	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	D1+	PBS1	—	AN	USB DAC0 output
	AN2	PBS1	AN	—	A/D Converter external input channel
PB7/D1-/AN3	PB7	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	D1-	PBS1	—	AN	USB DAC1 output
	AN3	PBS1	AN	—	A/D Converter external input channel
PC0/D2+/AN4	PC0	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	D2+	PCS0	—	AN	USB DAC0 output
	AN4	PCS0	AN	—	A/D Converter external input channel
PC1/D2-/AN5	PC1	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	D2-	PCS0	—	AN	USB DAC1 output
	AN5	PCS0	AN	—	A/D Converter external input channel
PC2/VDDIO/STPI/STPB	PC2	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	VDDIO	PCS0 PMPS	PWR	—	Multi-power input
	STPI	PCS0	ST	—	STM capture input
	STPB	PCS0	—	CMOS	STM inverted output
PC3/STCK/STP/AN6/VREF	PC3	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	STCK	PCS0	ST	—	STM clock input
	STP	PCS0	—	CMOS	STM output
	AN6	PCS0	AN	—	A/D Converter external input channel
	VREF	PCS0	AN	—	External reference voltage input
PC4/INT0/PTCK2/ PTP3B/AN7/VREFI	PC4	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	INT0	PCS1 INTEG INTC0	ST	—	External Interrupt 0 input
	PTCK2	PCS1	ST	—	PTM2 clock input
	PTP3B	PCS1	—	CMOS	PTM3 inverted output
	AN7	PCS1	AN	—	A/D Converter external input channel
	VREFI	PCS1	AN	—	A/D Converter PGA input
PC5/PTP2I/PTP3/AN8	PC5	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	PTP2I	PCS1	ST	—	PTM2 capture input
	PTP3	PCS1	—	CMOS	PTM3 output
	AN8	PCS1	AN	—	A/D Converter external input channel
PC6/INT1/PTCK3/ PTP2B/SCLM/AN9	PC6	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	INT1	PCS1 INTEG INTC1	ST	—	External Interrupt 1 input
	PTCK3	PCS1	ST	—	PTM3 clock input
	PTP2B	PCS1	—	CMOS	PTM2 inverted output
	SCLM	PCS1 IFS	—	NMOS	I^2C clock line
	AN9	PCS1	AN	—	A/D Converter external input channel

Pin Name	Function	OPT	I/T	O/T	Description
PC7/PTP3I/PTP2/ SDAM/AN10	PC7	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	PTP3I	PCS1	ST	—	PTM3 capture input
	PTP2	PCS1	—	CMOS	PTM2 output
	SDAM	PCS1 IFS	ST	NMOS	I²C data line
	AN10	PCS1	AN	—	A/D Converter external input channel
SVBUS	SVBUS	—	AN	—	USB VBUS connector
CC1	CC1	—	AN	AN	USB Type-C PD configuration channel 1
CC2	CC2	—	AN	AN	USB Type-C PD configuration channel 2
HVO1	HVO1	—	—	OD	Open drain output 1
HVO2	HVO2	—	—	OD	Open drain output 2
VDD	VDD	—	PWR	—	Positive power supply
VSS	VSS	—	PWR	—	Negative power supply, ground
DVDD/AVDD	DVDD	—	PWR	—	Digital positive power supply
	AVDD	—	PWR	—	Analog positive power supply
DVSS/AVSS	DVSS	—	PWR	—	Digital negative power supply
	AVSS	—	PWR	—	Analog negative power supply
VDD1P8	VDD1P8	—	PWR	AN	Regulator output pin and 1.8V Digital positive power supply
VCONN	VCONN	—	PWR	—	VCONN power 3V~5.5V input

Legend: I/T: Input type; O/T: Output type; OPT: Optional by register option;
CMOS: CMOS output; NMOS: NMOS output; ST: Schmitt Trigger input;
AN: Analog signal; OD: Open-Drain output; PWR: Power.

Interconnection Signal Description

Several signals listed in the following table are not connected to external package pins. These signals are interconnection lines between the MCU and the PD PHY. Users should properly configure the relevant control bits to implement correct interconnection.

MCU Signal Name	PD PHY Signal Name	Function	Description
PA4	HVI1	PA4	General purpose I/O. Register enabled pull-up and wake-up Internally connected to PD PHY HVI1
		HVI1	Open drain control logic 1 Internally connected to MCU PA4
PA5	HVI2	PA5	General purpose I/O. Register enabled pull-up and wake-up Internally connected to PD PHY HVI2
		HVI2	Open drain control logic 2 Internally connected to MCU PA5
PB1	SYSRESETN	PB1	General purpose I/O. Register enabled pull-up Internally connected to PD PHY SYSRESETN
		SYSRESETN	PD PHY reset input signal Internally connected to MCU PB1
PB2/INT2	BUS_INT	PB2	General purpose I/O. Register enabled pull-up Internally connected to PD PHY BUS_INT
		INT2	External Interrupt 2 input Internally connected to PD PHY BUS_INT
		BUS_INT	Bus interrupt output signal Internally connected to MCU PB2/INT2

MCU Signal Name	PD PHY Signal Name	Function	Description
PD0/SCLM	SCL_PD	PD0	General purpose I/O. Register enabled pull-up Internally connected to PD PHY SCL_PD
		SCLM	I²C clock line (Master mode) Internally connected to PD PHY SCL_PD
		SCL_PD	I²C clock line (Slave mode) Internally connected to MCU PD0/SCLM
PD1/SDAM	SDA_PD	PD1	General purpose I/O. Register enabled pull-up Internally connected to PD PHY SDA_PD
		SDAM	I²C data line (Master mode) Internally connected to PD PHY SDA_PD
		SDA_PD	I²C data line (Slave mode) Internally connected to MCU PD1/SDAM

Absolute Maximum Ratings

Supply Voltage ..... V_{SS}-0.3V to 6.0V

Input Voltage ..... V_{SS}-0.3V to V_{DD}+0.3V

Input Voltage for CC1/CC2/HVO1/HVO2 ..... 0V to 26V

Storage Temperature ..... -60°C to 150°C

Operating Temperature ..... -40°C to 85°C

I_{OH} Total ..... -80mA

I_{OL} Total 80mA

Total Power Dissipation 500mW

Note: These are stress ratings only. Stresses exceeding the range specified under "Absolute Maximum Ratings" may cause substantial damage to the device. Functional operation of the device at other conditions beyond those listed in the specification is not implied and prolonged exposure to extreme conditions may affect device reliability.

D.C. Characteristics

For data in the following tables, note that factors such as oscillator type, operating voltage, operating frequency, pin load conditions, temperature and program instruction type, etc., can all exert an influence on the measured values.

Operating Voltage Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_{DD}	Operating Voltage – HIRC	CKS[2:0]=010B: $f_H/4$, f_H=12MHz f_{SYS}=3MHz	2.2	—	5.5	V
		$f_{SYS}=f_{HIRC}$=12MHz	2.7	—	5.5	V
		$f_{SYS}=f_{HIRC}$=16MHz	3.3	—	5.5	V
		$f_{SYS}=f_{HIRC}$=20MHz	4.0	—	5.5	V
	Operating Voltage – LIRC	f_{SYS}=32kHz	2.2	—	5.5	V

Operating Current Characteristics

Ta=-40°C~85°C

Symbol	Operating Mode	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{DD}	SLOW Mode – LIRC	2.2V	f_{SYS}=32kHz	—	8	16	μA
		3V		—	10	20	
		5V		—	30	50	
	FAST Mode – HIRC	2.2V	CKS[2:0]=010B: f_H/4, f_H=12MHz, f_{SYS}=3MHz	—	0.2	0.4	mA
		3V		—	0.3	0.5	
		5V		—	0.7	1.1	
		2.7V	f_{SYS}=f_{HIRC}=12MHz	—	1.0	1.4	mA
		3V		—	1.2	1.8	
		5V		—	2.4	3.6	
		3.3V	f_{SYS}=f_{HIRC}=16MHz	—	1.5	3.0	mA
		5V		—	2.5	5.0	
		4.0V	f_{SYS}=f_{HIRC}=20MHz	—	5.4	7.2	mA
		5V		—	9	12	

Note: When using the characteristic table data, the following notes should be taken into consideration:

1. Any digital inputs are setup in a non floating condition.
2. All measurements are taken under conditions of no load and with all peripherals in an off state.
3. There are no DC current paths.
4. All Operating Current values are measured using a continuous NOP instruction program loop.

Standby Current Characteristics

Ta=25°C, unless otherwise specified

Symbol	Standby Mode	Test Conditions		Min.	Typ.	Max.	Max. @85°C	Unit
		V_{DD}	Conditions					
I_{STB}	SLEEP Mode	2.2V	WDT off	—	0.45	0.80	7.00	μA
		3V		—	0.45	0.90	8.00	
		5V		—	0.5	2.0	10.0	
		2.2V	WDT on	—	1.2	2.4	2.9	μA
		3V		—	1.5	3.0	3.6	
		5V		—	3	5	6	
	IDLE0 Mode – LIRC	2.2V	f_{SUB} on	—	2.4	4.0	4.6	μA
		3V		—	3.0	5.0	5.7	
		5V		—	5	10	11	
	IDLE1 Mode – HIRC	2.2V	f_{SUB} on, CKS[2:0]=010B: f_H/4, f_H=12MHz, f_{SYS}=3MHz	—	240	300	300	μA
		3V		—	375	450	450	
		5V		—	750	900	900	
		2.7V	f_{SUB} on, f_{SYS}=f_{HIRC}=12MHz	—	432	600	720	μA
		3V		—	540	750	900	
		5V		—	800	1200	1440	
		3.3V	f_{SUB} on, f_{SYS}=f_{HIRC}=16MHz	—	0.80	1.20	1.44	mA
		5V		—	1.4	2.0	2.4	
		4.0V	f_{SUB} on, f_{SYS}=f_{HIRC}=20MHz	—	1.32	1.98	2.37	mA
		5V		—	2.50	3.30	3.96	

Note: When using the characteristic table data, the following notes should be taken into consideration:

1. Any digital inputs are setup in a non floating condition.
2. All measurements are taken under conditions of no load and with all peripherals in an off state.
3. There are no DC current paths.
4. All Standby Current values are taken after a HALT instruction execution thus stopping all instruction execution.

A.C. Characteristics

For data in the following tables, note that factors such as oscillator type, operating voltage, operating frequency and temperature etc., can all exert an influence on the measured values.

Internal High Speed Oscillator – HIRC – Frequency Accuracy

During the program writing operation the writer will trim the HIRC oscillator at a user selected HIRC frequency and user selected voltage of either 3V or 5V.

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{HIRC}	12MHz Writer Trimmed HIRC Frequency	3V/5V	25°C	-1%	12	+1%	MHz
			-40°C~85°C	-2%	12	+2%	
		2.7V~5.5V	25°C	-2.5%	12	+2.5%	
			-40°C~85°C	-3%	12	+3%	
		2.2V~5.5V	25°C	-20%	12	+20%	
			-40°C~85°C	-30%	12	+30%	
	16MHz Writer Trimmed HIRC Frequency	5V	25°C	-1%	16	+1%	MHz
			-40°C~85°C	-2%	16	+2%	
		3.3V~5.5V	25°C	-2.5%	16	+2.5%	
			-40°C~85°C	-3%	16	+3%	
	20MHz Writer Trimmed HIRC Frequency	5V	25°C	-1%	20	+1%	MHz
			-40°C~85°C	-2%	20	+2%	
		4.0V~5.5V	25°C	-2.5%	20	+2.5%	
			-40°C~85°C	-3%	20	+3%	

Note: 1. The 3V/5V values for VDD are provided as these are the two selectable fixed voltages at which the HIRC frequency is trimmed by the writer.

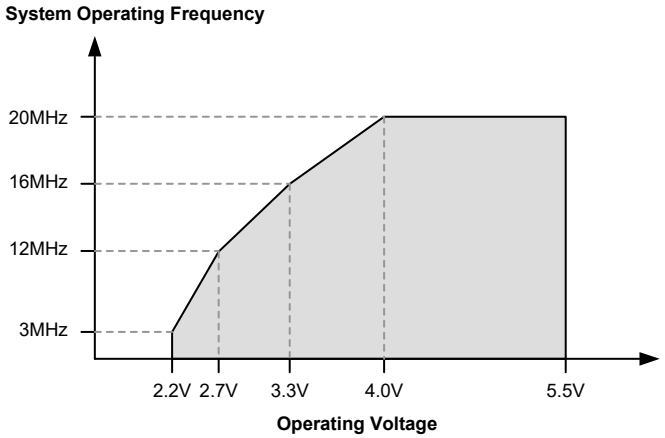
2. The row below the 3V/5V trim voltage row is provided to show the values for the full V_{DD} range operating voltage. It is recommended that the trim voltage is fixed at 3V for application voltage ranges from 2.2V to 3.6V and fixed at 5V for application voltage ranges from 3.3V to 5.5V.

3. The minimum and maximum tolerance values provided in the table are only for the frequency at which the writer trims the HIRC oscillator. After trimming at this chosen specific frequency any change in HIRC oscillator frequency using the oscillator register control bits by the application program will give a frequency tolerance to within ±20%.

Internal Low Speed Oscillator Characteristics – LIRC

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{LIRC}	LIRC Frequency	3V	25°C	-2%	32	+2%	kHz
		2.2V~5.5V	-40°C~85°C	-7%	32	+7%	
t_{START}	LIRC Start Up Time	—	-40°C~85°C	—	—	100	μs

Operating Frequency Characteristic Curves



System Start Up Time Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
t_{SST}	System Start-up Time (Wake-up from Condition where f_{SYS} is off)	—	$f_{SYS}=f_H\sim f_H/64$, $f_H=f_{HIRC}$	—	16	—	t_{HIRC}
			$f_{SYS}=f_{SUB}=f_{LIRC}$	—	2	—	t_{LIRC}
	System Start-up Time (Wake-up from Condition where f_{SYS} is on)	—	$f_{SYS}=f_H\sim f_H/64$, $f_H=f_{HIRC}$	—	2	—	t_H
			$f_{SYS}=f_{SUB}=f_{LIRC}$	—	2	—	t_{SUB}
	System Speed Switch Time (FAST to Slow Mode or SLOW to FAST Mode)	—	f_{HIRC} switches from off → on	—	16	—	t_{HIRC}
t_{RSTD}	System Reset Delay Time (Reset Source from Power-on Reset or LVR Hardware Reset)	—	RR_{POR}=5V/ms	14	16	18	ms
	System Reset Delay Time (LVRC/WDTC Software Reset)	—	—				
	System Reset Delay Time (Reset Source from WDT Overflow Reset)	—	—				

Note: 1. For the System Start-up time values, whether f_{SYS} is on or off depends upon the mode type and the chosen f_{SYS} system oscillator. Details are provided in the System Operating Modes section.

2. The time units, shown by the symbols t_{HIRC} etc. are the inverse of the corresponding frequency values as provided in the frequency tables. For example $t_{HIRC}=1/f_{HIRC}$, $t_{SYS}=1/f_{SYS}$ etc.

3. If the LIRC is used as the system clock and if it is off when in the SLEEP Mode, then an additional LIRC start up time, t_{START}, as provided in the LIRC frequency table, must be added to the t_{SST} time in the table above.

4. The System Speed Switch Time is effectively the time taken for the newly activated oscillator to start up.

Input/Output Characteristics

Input/Output (without Multi-power) D.C. Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{IL}	Input Low Voltage for I/O Ports (Except PB4~PB5 Pins)	5V	—	0	—	1.5	V
		—	—	0	—	$0.2V_{DD}$	
V_{IH}	Input High Voltage for I/O Ports (Except PB4~PB5 Pins)	5V	—	3.5	—	5.0	V
		—	—	$0.8V_{DD}$	—	V_{DD}	
I_{OL}	Sink Current for I/O Ports (Except PB4~PB5 Pins)	3V	$V_{OL}=0.1V_{DD}$	16	32	—	mA
		5V		32	65	—	
I_{OH}	Source Current for I/O Ports (Except PB4~PB5 Pins)	3V	$V_{OH}=0.9V_{DD}$, SLEDCn[m+1:m]=00B (n=0~1; m=0, 2, 4, 6)	-0.7	-1.5	—	mA
		5V		-1.5	-2.9	—	
		3V	$V_{OH}=0.9V_{DD}$, SLEDCn[m+1:m]=01B (n=0~1; m=0, 2, 4, 6)	-1.3	-2.5	—	
		5V		-2.5	-5.1	—	
		3V	$V_{OH}=0.9V_{DD}$, SLEDCn[m+1:m]=10B (n=0~1; m=0, 2, 4, 6)	-1.8	-3.6	—	
		5V		-3.6	-7.3	—	
		3V	$V_{OH}=0.9V_{DD}$, SLEDCn[m+1:m]=11B (n=0~1; m=0, 2, 4, 6)	-4	-8	—	
		5V		-8	-16	—	
R_{PH}	Pull-high Resistance for I/O Ports (1) (Except PB4~PB5 Pins)	3V	—	20	60	100	kΩ
		5V		10	30	50	
I_{LEAK}	Input leakage current for I/O Ports (Except PB4~PB5 Pins)	5V	$V_{IN}=V_{DD}$ or $V_{IN}=V_{SS}$	—	—	±1	μA
t_{INT}	Interrupt Input Pin Minimum Pulse Width	—	—	10	—	—	μs
t_{TCK}	TM Clock Input Minimum Pulse Width	—	—	0.3	—	—	μs
t_{TPI}	TM Capture Input Minimum Pulse Width	—	—	0.3	—	—	μs
f_{TMCLK}	TM Maximum Timer Clock Source Frequency	5V	—	—	—	1	f_{SYS}
t_{CPW}	TM Minimum Capture Pulse Width	—	—	t_{CPW}(2)	—	—	μs
t_{SRESET}	Minimum Software Reset Pulse Width to Reset	—	—	45	90	120	μs

Note: 1. The R_{PH} internal pull-high resistance value is calculated by connecting to ground and enabling the input pin with a pull-high resistor and then measuring the pin current at the specified supply voltage level. Dividing the voltage by this measured current provides the R_{PH} value.

2. For PTMn:

If PTnCAPTS=0, then $t_{CPW}=\max(2\times t_{TMCLK}, t_{TPI})$

If PTnCAPTS=1, then $t_{CPW}=\max(2\times t_{TMCLK}, t_{TCK})$

Ex1: If PTnCAPTS=0, f_{TMCLK}=16MHz, t_{TPI}=0.3μs, then $t_{CPW}=\max(0.125\mu s, 0.3\mu s)=0.3\mu s$

Ex2: If PTnCAPTS=1, f_{TMCLK}=16MHz, t_{TCK}=0.3μs, then $t_{CPW}=\max(0.125\mu s, 0.3\mu s)=0.3\mu s$

Ex3: If PTnCAPTS=0, f_{TMCLK}=8MHz, t_{TPI}=0.3μs, then $t_{CPW}=\max(0.25\mu s, 0.3\mu s)=0.3\mu s$

Ex4: If PTnCAPTS=0, f_{TMCLK}=4MHz, t_{TPI}=0.3μs, then $t_{CPW}=\max(0.5\mu s, 0.3\mu s)=0.5\mu s$

For STM:

$t_{CPW}=\max(2\times t_{TMCLK}, t_{TPI})$

Ex1: If f_{TMCLK}=16MHz, t_{TPI}=0.3μs, then $t_{CPW}=\max(0.125\mu s, 0.3\mu s)=0.3\mu s$

Ex2: If f_{TMCLK}=8MHz, t_{TPI}=0.3μs, then $t_{CPW}=\max(0.25\mu s, 0.3\mu s)=0.3\mu s$

Ex3: If f_{TMCLK}=4MHz, t_{TPI}=0.3μs, then $t_{CPW}=\max(0.5\mu s, 0.3\mu s)=0.5\mu s$

Where $t_{TMCLK}=1/f_{TMCLK}$

Input/Output (with Multi-power) D.C. Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	V_{DD} Power Supply for PB4~PB5 Pins	—	—	2.2	5.0	5.5	V
V_{DDIO}	V_{DDIO} Power Supply for PB4~PB5 Pins	—	—	2.2	—	V_{DD}	V
V_{IL}	Input Low Voltage for PB4~PB5 Pins	5V	Pin power=V_{DD} or V_{DDIO}, V_{DDIO}=V_{DD}	0	—	1.5	V
		—	Pin power=V_{DD} or V_{DDIO}	0	—	0.2(V_{DD}/V_{DDIO})	
V_{IH}	Input High Voltage for PB4~PB5 Pins	5V	Pin power=V_{DD} or V_{DDIO}, V_{DDIO}=V_{DD}	3.5	—	5.0	V
		—	Pin power=V_{DD} or V_{DDIO}	0.8(V_{DD}/V_{DDIO})	—	V_{DD}/V_{DDIO}	
I_{OL}	Sink Current for PB4~PB5 Pins	3V	V_{OL}=0.1(V_{DD} or V_{DDIO}), V_{DDIO}=V_{DD}	16	32	—	mA
		5V		32	65	—	
		5V	V_{OL}=0.1(V_{DD} or V_{DDIO}), V_{DDIO}=3V	20	40	—	
I_{OH}	Source Current for PB4~PB5 Pins	3V	V_{OH}=0.9(V_{DD} or V_{DDIO}), V_{DDIO}=V_{DD} SLEDC0[7:6]=00B	-0.7	-1.5	—	mA
		5V		-1.5	-2.9	—	
		5V	V_{OH}=0.9(V_{DD} or V_{DDIO}), V_{DDIO}=3V SLEDC0[7:6]=00B	-0.40	-0.85	—	
		3V	V_{OH}=0.9(V_{DD} or V_{DDIO}), V_{DDIO}=V_{DD} SLEDC0[7:6]=01B	-1.3	-2.5	—	
		5V		-2.5	-5.1	—	
		5V	V_{OH}=0.9(V_{DD} or V_{DDIO}), V_{DDIO}=3V, SLEDC0[7:6]=01B	-0.70	-1.35	—	
		3V	V_{OH}=0.9(V_{DD} or V_{DDIO}), V_{DDIO}=V_{DD} SLEDC0[7:6]=10B	-1.8	-3.6	—	
		5V		-3.6	-7.3	—	
		5V	V_{OH}=0.9(V_{DD} or V_{DDIO}), V_{DDIO}=3V SLEDC0[7:6]=10B	-0.95	-1.90	—	
		3V	V_{OH}=0.9(V_{DD} or V_{DDIO}), V_{DDIO}=V_{DD} SLEDC0[7:6]=11B	-4	-8	—	
		5V		-8	-16	—	
		5V	V_{OH}=0.9(V_{DD} or V_{DDIO}), V_{DDIO}=3V SLEDC0[7:6]=11B	-2.5	-5.0	—	
R_{PH}	Pull-high Resistance for PB4~PB5 Pins (Note)	3V	Pin power=V_{DD} or V_{DDIO}, V_{DDIO}=V_{DD}	20	60	100	kΩ
		5V		10	30	50	
		5V	Pin power=V_{DD} or V_{DDIO}, V_{DDIO}=3V	36	110	180	
I_{LEAK}	Input Leakage Current for PB4~PB5 Pins	5V	V_{IN}=V_{SS} or V_{IN}=V_{DD} or V_{DDIO}	—	—	±1	μA

Note: The R_{PH} internal pull-high resistance value is calculated by connecting to ground and enabling the input pin with a pull-high resistor and then measuring the pin current at the specified supply voltage level. Dividing the voltage by this measured current provides the R_{PH} value.

Memory Electrical Characteristics

Ta=-40°C~85°C, unless otherwise specified

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	V_{DD} for Read/Write	—	—	2.2	—	5.5	V
Flash Program Memory							
t_{FWR}	Write Time	—	FWERTS=0	—	2.2	2.7	ms
		—	FWERTS=1	—	3.0	3.6	

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
t_{FER}	Erase Time	—	FWERTS=0	—	3.2	3.9	ms
		—	FWERTS=1	—	3.7	4.5	
E_P	Cell Endurance	—	—	100K	—	—	E/W
t_{RETD}	Data Retention Time	—	Ta=25°C	—	40	—	Year
t_{ACTV}	ROM Activation Time – Wake-up from Power Down Mode	—	—	32	—	64	μs
Data EEPROM Memory							
t_{EERD}	Read Time	—	—	—	—	4	t_{SYS}
t_{EEWR}	Write Time (Byte Mode)	—	EWERTS=0	—	5.4	6.6	ms
		—	EWERTS=1	—	6.7	8.1	
	Write Time (Page Mode)	—	EWERTS=0	—	2.2	2.7	
		—	EWERTS=1	—	3.0	3.6	
t_{EEER}	Erase Time	—	EWERTS=0	—	3.2	3.9	ms
		—	EWERTS=1	—	3.7	4.5	
E_P	Cell Endurance	—	—	100K	—	—	E/W
t_{RETD}	Data Retention Time	—	Ta=25°C	—	40	—	Year
RAM Data Memory							
V_{DR}	RAM Data Retention Voltage	—	—	1.0	—	—	V

Note: 1. The ROM activation time t_{ACTV} should be added when calculating the total system start-up time of a wake-up from the power down mode.
2. "E/W" means Erase/Write times.

LVD/LVR Electrical Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	2.2	—	5.5	V
V_{LVR}	Low Voltage Reset Voltage	—	LVR enabled, voltage select 2.1V	-3%	2.1	+3%	V
			LVR enabled, voltage select 2.55V		2.55		
			LVR enabled, voltage select 3.15V		3.15		
			LVR enabled, voltage select 3.8V		3.8		
V_{LVD}	Low Voltage Detect Voltage	—	LVD enabled, voltage select 2.0V	-5%	2.0	+5%	V
			LVD enabled, voltage select 2.2V		2.2		
			LVD enabled, voltage select 2.4V		2.4		
			LVD enabled, voltage select 2.7V		2.7		
			LVD enabled, voltage select 3.0V		3.0		
			LVD enabled, voltage select 3.3V		3.3		
			LVD enabled, voltage select 3.6V		3.6		
			LVD enabled, voltage select 4.0V		4.0		
I_{LVRLVD}	Operating Current	3V	LVR enable, LVD enable, V_{LVR}=2.1V, V_{LVD}=2.0V	—	—	10	μA
		5V		—	10	15	
t_{LVDS}	LVDO Stable Time	—	For LVR enable, LVD off → on	—	—	18	μs
		—	For LVR disable, LVD off → on	—	—	150	
t_{LVR}	Minimum Low Voltage Width to Reset	—	—	120	240	480	μs

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
t_{LVD}	Minimum Low Voltage Width to Interrupt	—	—	60	120	240	μs
I_{LVR}	Additional Current Consumption for LVR Enable	5V	LVD disabled	—	—	14	μA
I_{LVD}	Additional Current Consumption for LVD Enable	5V	LVR disabled	—	—	14	μA

A/D Converter Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	2.2	—	5.5	V
V_{ADI}	Input Voltage	—	—	0	—	V_{REF}	V
V_{REF}	Reference Voltage	—	—	2.2	—	V_{DD}	V
N_R	Resolution	—	—	—	—	12	Bit
DNL	Differential Non-linearity	2.2V	SAINS[3:0]=0000B, SAVRS[1:0]=01B, V_{REF}=V_{DD}, t_{ADCK}=0.5μs	-3	—	+3	LSB
		3V					
		5V					
		2.2V	SAINS[3:0]=0000B, SAVRS[1:0]=01B, V_{REF}=V_{DD}, t_{ADCK}=10μs				
		3V					
		5V					
INL	Integral Non-linearity	2.2V	SAINS[3:0]=0000B, SAVRS[1:0]=01B, V_{REF}=V_{DD}, t_{ADCK}=0.5μs	-4	—	+4	LSB
		3V					
		5V					
		2.2V	SAINS[3:0]=0000B, SAVRS[1:0]=01B, V_{REF}=V_{DD}, t_{ADCK}=10μs				
		3V					
		5V					
I_{ADC}	Additional Current Consumption for A/D Converter Enable	2.2V	No load, t_{ADCK}=2.0μs	—	280	400	μA
		3V	No load, t_{ADCK}=0.5μs	—	450	600	
		5V		—	850	1000	
t_{ADCK}	Clock Period	—	2.2V≤V_{DD}≤5.5V	0.5	—	10.0	μs
t_{ON2ST}	A/D Converter On-to-Start Time	—	—	4	—	—	μs
t_{ADS}	Sampling Time	—	—	—	4	—	t_{ADCK}
t_{ADC}	Conversion Time (Including A/D Sample and Hold Time)	—	—	—	16	—	t_{ADCK}
I_{PGA}	Additional Current Consumption for PGA Enable	2.2V	No load, PGAIS=1, PGAGS[1:0]=01	—	500	1000	μA
		3V		—	600	1200	
		5V		—	700	1400	
V_{CM}	PGA Common Mode Voltage Range	3V	—	V_{SS}+0.1	—	V_{DD}-1.4	V
		5V		V_{SS}+0.1	—	V_{DD}-1.4	
V_{OR}	PGA Maximum Output Voltage Range	2.2V	—	V_{SS}+0.1	—	V_{DD}-0.1	V
		3V		V_{SS}+0.1	—	V_{DD}-0.1	
		5V		V_{SS}+0.1	—	V_{DD}-0.1	
V_{VR}	PGA Fix Output Voltage	2.2V~5.5V	V_{RI}=V_{BGREF} (PGAIS=1)	-1%	2	+1%	V
		3.2V~5.5V		-1%	3	+1%	
		4.2V~5.5V		-1%	4	+1%	

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{IR}	PGA Input Voltage Range	3V	Gain=1, PGAIS=0, Relative gain, Gain error <±5%	V_{SS}+0.1	—	V_{DD}-1.4	V
		5V		V_{SS}+0.1	—	V_{DD}-1.4	

Internal Reference Voltage Electrical Characteristics

Ta=-40°C~85°C, unless otherwise specified

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	2.2	—	5.5	V
V_{BGREF}	Bandgap Reference Voltage	2.2V~5.5V	—	-1%	1.2	+1%	V
I_{BGREF}	Operating Current	5.5V	—	—	25	35	μA
PSRR	Power Supply Rejection Ratio	—	Ta=25°C, V_{RIPPLE}=1V_{P-P}, f_{RIPPLE}=100Hz	75	—	—	dB
En	Output Noise	—	Ta=25°C, no load current, f=0.1Hz~10Hz	—	300	—	μV_{RMS}
I_{SD}	Shutdown Current	—	VBGREN=0	—	—	0.1	μA
t_{START}	Start Up Time	2.2V~5.5V	Ta=25°C	—	—	400	μs

Note: 1. All the above parameters are measured under conditions of no load condition unless otherwise described.
2. A 0.1μF ceramic capacitor should be connected between VDD and GND.
3. The V_{BGREF} voltage is used as the A/D converter PGA reference voltage input.

Operational Amplifier Electrical Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{OPA}	Additional Current for OPAMP Enable	3V	No load	—	150	400	μA
		5V		—	300	600	
V_{OS}	Input Offset Voltage	3V	Without calibration (OPOOF[5:0]=100000B)	-15	—	15	mV
		5V		-15	—	15	
		3V	With calibration	-2	—	2	
		5V		-2	—	2	
V_{CM}	Common Mode Voltage Range	5V	—	V_{SS}	—	V_{DD}-1.4	V
V_{OR}	Maximum Output Voltage Range	5V	—	V_{SS}+0.1	—	V_{DD}-0.1	V
R_{OPAR1}	Value of OPAR1	5V	—	7.5	10.0	12.5	kΩ
SR	Slew Rate	5V	No load	0.6	1.8	—	V/μs
GBW	Gain Bandwidth	5V	R_{LOAD}=1MΩ, C_{LOAD}=100pF	600	2200	—	kHz
PSRR	Power Supply Rejection Ratio	5V	—	60	80	—	dB
CMRR	Common Mode Rejection Ratio	5V	—	60	80	—	dB
Ga	PGA Gain Accuracy (Note)	3V	Relative gain	-5	—	5	%
		5V		-5	—	5	

Note: The PGA gain accuracy is guaranteed only when the PGA output voltage meets the V_{OR} specification.

USB Auto Detector Electrical Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DAC}	DAC Operating Voltage	—	—	2.2	—	5.5	V
I_{DAC}	DAC Operating Current	3V	No load	—	0.6	0.9	mA
		5V		—	1.0	1.5	mA
I_{DACSD}	DAC Shutdown Current	—	No load	—	—	0.1	μA
N_R	DAC Resolution	—	—	—	8	—	Bit
DNL	DAC Differential Non-linearity	—	No load, DAC reference=V_{DD}	—	—	±1	LSB
INL	DAC Integral Non-linearity	—	No load, DAC reference=V_{DD}	—	—	±2	LSB
V_{DACO}	Output Voltage Range	—	Code=00H	V_{SS}	—	V_{SS}+0.2	V
		—	Code=FFH	V_{REF}-0.2	—	V_{REF}	V
V_{REF}	Reference Voltage	—	—	2	—	V_{DD}	V
t_{ST}	Settling Time	3V	C_{LOAD}=50pF	—	—	5	μs
		5V		—	—	5	
I_{DACOL}	Output Sink Current	3V	Data word=00H, V_{DACO}=0.1V_{REF}, V_{REF}=V_{DD}	20	—	—	μA
		5V		40	—	—	
I_{DACOH}	Output Source Current	3V	Data word=FFH, V_{DACO}=0.9V_{REF}, V_{REF}=V_{DD}	-20	—	—	μA
		5V		-40	—	—	
I_{SC}	Output Short-Circuit Current	3V	Data word=FFH	-0.25	—	—	mA
		5V		-0.4	—	—	
R_{ON}	Analog Switch On Resistance between D1+/D2+ and D1-/D2-	5V	—	—	20	35	Ω
R_{PL}	Pull-Low Resistance for D1+, D2+	5V	—	400	700	1400	kΩ
	Pull-Low Resistance for D1-, D2-	5V	—	15	20	23	kΩ
ERR	The Error for D1+, D1-, D2+, D2- Output Voltage	5V	DAC reference=V_{DD}, DAC digital value=148, D1+, D1-, D2+ or D2- connect 150kΩ to ground	2.57	2.70	2.84	V
		5V	DAC reference=V_{DD}, DAC digital value=110, D1+, D1-, D2+ or D2- connect 150kΩ to ground	1.9	2.0	2.1	V

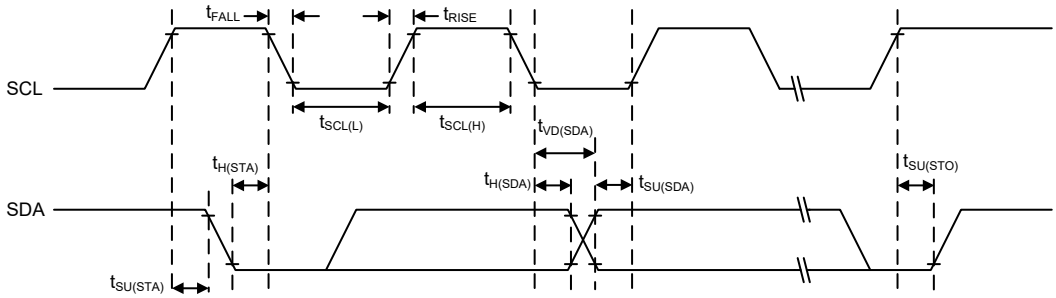
SIM I²C Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{I2C}	I²C Standard Mode (100kHz) f_{SYS} Frequency (Note)	—	No clock debounce	2	—	—	MHz
			2 system clock debounce	4	—	—	
			4 system clock debounce	4	—	—	
	I²C Fast Mode (400kHz) f_{SYS} Frequency (Note)	—	No clock debounce	4	—	—	MHz
			2 system clock debounce	8	—	—	
			4 system clock debounce	8	—	—	
f_{SCL}	SCL Clock Frequency	3V/5V	Standard mode	—	—	100	kHz
			Fast mode	—	—	400	
$t_{SCL(H)}$	SCL Clock High Time	3V/5V	Standard mode	3.5	—	—	μs
			Fast mode	0.9	—	—	

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
$t_{SCL(L)}$	SCL Clock Low Time	3V/5V	Standard mode	3.5	—	—	μs
			Fast mode	0.9	—	—	
t_{FALL}	SCL and SDA Fall Time	3V/5V	Standard mode	—	—	1.3	μs
			Fast mode	—	—	0.34	
t_{RISE}	SCL and SDA Rise Time	3V/5V	Standard mode	—	—	1.3	μs
			Fast mode	—	—	0.34	
$t_{SU(SDA)}$	SDA Data Setup Time	3V/5V	Standard mode	0.25	—	—	μs
			Fast mode	0.1	—	—	
$t_{H(SDA)}$	SDA Data Hold Time	3V/5V	—	0.1	—	—	μs
$t_{VD(SDA)}$	SDA Data Valid Time	3V/5V	—	—	—	0.6	μs
$t_{SU(STA)}$	Start Condition Setup Time	3V/5V	Standard mode	3.5	—	—	μs
			Fast mode	0.6	—	—	
$t_{H(STA)}$	Start Condition Hold Time	3V/5V	Standard mode	4.0	—	—	μs
			Fast mode	0.6	—	—	
$t_{SU(STO)}$	Stop Condition Setup Time	3V/5V	Standard mode	3.5	—	—	μs
			Fast mode	0.6	—	—	

Note: Using the debounce function can make the transmission more stable and reduce the probability of communication failure due to interference.

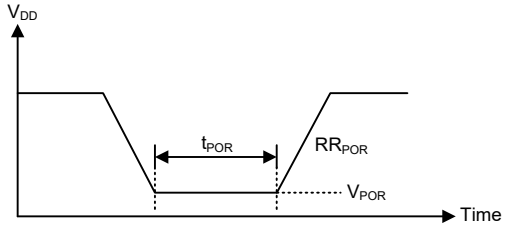


I²C Timing Diagram

Power-on Reset Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{POR}	V_{DD} Start Voltage to Ensure Power-on Reset	—	—	—	—	100	mV
RR_{POR}	V_{DD} Rising Rate to Ensure Power-on Reset	—	—	0.035	—	—	V/ms
t_{POR}	Minimum Time for V_{DD} Stays at V_{POR} to Ensure Power-on Reset	—	—	1	—	—	ms



PD PHY Operating Voltage Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_{DVDD_PD}	Operating Voltage	—	2.6	—	5.5	V
V_{AVDD_PD}	Operating Voltage	—	2.6	—	5.5	V

PD PHY Operating Current Characteristics

Ta=-40°C~85°C

Symbol	Operating Mode	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD_PD}	Conditions				
I_{DD_PD}	Normal Mode	3V	PWR[3:0]=1101b	—	1.5	3.0	mA
		5V		—	2	5	mA
	SLEEP1 Mode	3V	PWR[3:0]=0010b	—	10	20	μA
		5V		—	20	60	μA
	SLEEP0 Mode	3V	PWR[3:0]=0000b	—	3	6	μA
		5V		—	5	45	μA

PD PHY LDO Electrical Characteristics

$V_{DD\_PD}=V_{IN}$, $V_{IN}=V_{OUT}+0.8V$, $C_{LOAD}=4.7\mu F$, Ta=-40°C~85°C, unless otherwise specified

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD_PD}	Conditions				
V_{DD_PD}	Supply Voltage	—	—	2.6	—	5.5	V
V_{OUT}	Output Voltage	—	Ta=25°C, I_{LOAD}=0.5mA	-4%	1.8	4%	V
		—	I_{LOAD}=0.5mA	-8%	1.8	8%	V
I_Q	Quiescent Current	5V	No load	—	—	10	μA
I_{OUT}	Output Current	—	V_{IN}=2.6V, ΔV_{OUT}=0.15V, V_{OUT}=1.8V	5	—	—	mA
TC	Temperature Coefficient	—	I_{LOAD}=5mA	—	±1.5	±2.0	mV/°C

PD PHY HVO Electrical Characteristics

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD_PD}	Conditions				
I_{OL}	Sink Current for HVO1, HVO2 Pins	5V	$V_{OL}=0.1V_{DD\_PD}$	20	35	—	mA

PD PHY Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_{VCONN}	VCONN Voltage	—	3.0	—	5.5	V
V_{CCOVP}	CC1, CC2 Over Voltage Protection	—	-6%	5	5%	V
I_{AVDD_PD}	AVDD Current While Waiting to Receive	—	—	35	—	μA
I_{RX}	AVDD Current While Receiving	CDR_SELECT=0	—	250	—	μA
I_{TX}	AVDD Current While Transmitting	—	—	—	1.5	mA
I_{CC1LK}	CC1 Pins Leakage Current	—	—	—	5	μA
I_{CC2LK}	CC2 Pins Leakage Current	—	—	—	5	μA
V_{REF}	DAC Reference Voltage	—	2.5	2.6	2.7	V

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
N_R	DAC Resolution	—	—	6	—	Bit
V_{DAC}	DAC Threshold	DAC[5:0]=000000b~111111b	-30	—	+30	mV
V_{SLC2}	Upper Slice Voltage Comparator Reference	—	800	850	880	mV
V_{SLC1}	Middle Slice Voltage Comparator Reference	—	520	550	590	mV
V_{SLC0}	Lower Slice Voltage Comparator Reference	—	220	250	280	mV
T_{SLCT}	Slice Analog Activity Test Timer Time	—	15	—	35	μs
V_{TH3}	3A CC Detect Threshold	—	1.12	1.16	1.25	V
V_{TH1P5}	1.5A CC Detect Threshold	—	580	610	640	mV
V_{TH_DEF}	Default Current CC Detect Threshold	—	160	200	250	mV
$V_{RDTH330}$	R_D Voltage Threshold for 330μA Pull Up	—	1.9	2.4	2.7	V
$V_{RDTH180}$	R_D Voltage Threshold for 180μA Pull Up	—	1.5	1.6	1.7	V
V_{RDTH80}	R_D Voltage Threshold for 80μA Pull Up	—	1.5	1.6	1.7	V
$V_{RATH330}$	R_A Voltage Threshold for 330μA Pull Up	—	760	800	830	mV
$V_{RATH180}$	R_A Voltage Threshold for 180μA Pull Up	—	280	320	420	mV
V_{RATH80}	R_A Voltage Threshold for 80μA Pull Up	—	180	200	225	mV
f_{I2C}	I^2C Clock Frequency	—	400	—	1000	kHz

PHY Power on Reset Electrical Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD_PD}	Conditions				
V_{POR_PD}	V_{DD_PD} Start Voltage to Ensure Power-on Reset	—	—	—	—	100	mV
RR_{POR_PD}	V_{DD_PD} Rising Rate to Ensure Power-on Reset	—	—	0.035	—	—	V/ms
t_{POR_PD}	Minimum Time for V_{DD_PD} Stays at V_{POR_PD} to Ensure Power-on Reset	—	—	1	—	—	ms

System Architecture

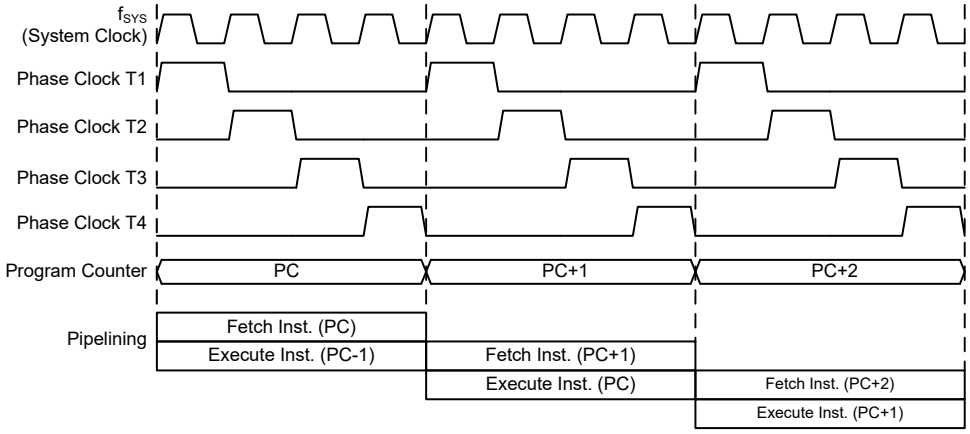
A key factor in the high-performance features of the Holtek range of microcontrollers is attributed to their internal system architecture. The device takes advantage of the usual features found within RISC microcontrollers providing increased speed of operation and enhanced performance. The pipelining scheme is implemented in such a way that instruction fetching and instruction execution are overlapped, hence instructions are effectively executed in one or two cycles for most of the standard or extended instructions respectively, with the exception of branch or call instructions which needs one more cycle. An 8-bit wide ALU is used in practically all instruction set operations, which carries out arithmetic operations, logic operations, rotation, increment, decrement, branch decisions, etc. The internal data path is simplified by moving data through the Accumulator and the ALU. Certain internal registers are implemented in the Data Memory and can be directly or indirectly addressed. The simple addressing methods of these registers along with additional architectural features ensure that a minimum of external components is required to provide a functional I/O and A/D control system with maximum reliability and flexibility. This makes the device suitable for low-cost, high-volume production for controller applications.

Clocking and Pipelining

The main system clock, derived from either an HIRC or LIRC oscillator is subdivided into four internally generated non-overlapping clocks, T1~T4. The Program Counter is incremented at the beginning of the T1 clock during which time a new instruction is fetched. The remaining T2~T4

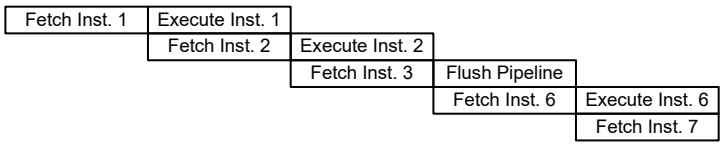
clocks carry out the decoding and execution functions. In this way, one T1~T4 clock cycle forms one instruction cycle. Although the fetching and execution of instructions takes place in consecutive instruction cycles, the pipelining structure of the microcontroller ensures that instructions are effectively executed in one instruction cycle. The exception to this are instructions where the contents of the Program Counter are changed, such as subroutine calls or jumps, in which case the instruction will take one more instruction cycle to execute.

For instructions involving branches, such as jump or call instructions, two machine cycles are required to complete instruction execution. An extra cycle is required as the program takes one cycle to first obtain the actual jump or call address and then another cycle to actually execute the branch. The requirement for this extra cycle should be taken into account by programmers in timing sensitive applications.



System Clocking and Pipelining

```
1          MOV A,[12H]
2          CALL DELAY
3          CPL [12H]
4          :
5          :
6  DELAY:  NOP
```



Instruction Fetching

Program Counter

During program execution, the Program Counter is used to keep track of the address of the next instruction to be executed. It is automatically incremented by one each time an instruction is executed except for instructions, such as “JMP” or “CALL” that demand a jump to a non-consecutive Program Memory address. For the device with a Program Memory capacity in excess of 8K words, the Program Memory high byte address must be setup by selecting a certain program memory bank which is implemented using the program memory bank pointer bit, PBP0. Only the lower 8 bits, known as the Program Counter Low Register, are directly addressable by the application program.

When executing instructions requiring jumps to non-consecutive addresses such as a jump instruction, a subroutine call, interrupt or reset, etc., the microcontroller manages program control by loading the required address into the Program Counter. For conditional skip instructions, once the condition has been met, the next instruction, which has already been fetched during the present instruction execution, is discarded and a dummy cycle takes its place while the correct instruction is obtained.

Program Counter	
High Byte	Low Byte (PCL)
PBP0, PC12~PC8	PCL7~PCL0

Program Counter

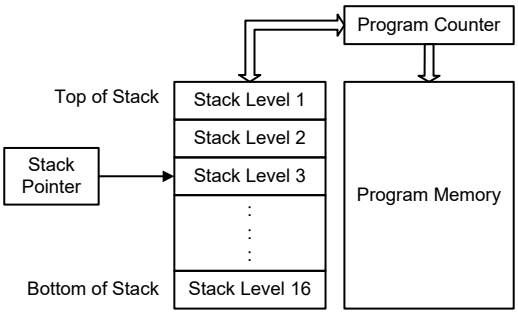
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly; however, as only this low byte is available for manipulation, the jumps are limited to the present page of memory that is 256 locations. When such program jumps are executed it should also be noted that a dummy cycle will be inserted. Manipulating the PCL register may cause program branching, so an extra cycle is needed to pre-fetch.

Stack

This is a special part of the memory which is used to save the contents of the Program Counter only. The stack has multiple levels and is neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the Stack Pointer, and is neither readable nor writeable. At a subroutine call or interrupt acknowledge signal, the contents of the Program Counter are pushed onto the stack. At the end of a subroutine or an interrupt routine, signaled by a return instruction, RET or RETI, the Program Counter is restored to its previous value from the stack. After a device reset, the Stack Pointer will point to the top of the stack.

If the stack is full and an enabled interrupt takes place, the interrupt request flag will be recorded but the acknowledge signal will be inhibited. When the Stack Pointer is decremented, by RET or RETI, the interrupt will be serviced. This feature prevents stack overflow allowing the programmer to use the structure more easily. However, when the stack is full, a CALL subroutine instruction can still be executed which will result in a stack overflow. Precautions should be taken to avoid such cases which might cause unpredictable program branching.

If the stack is overflow, the first Program Counter save in the stack will be lost.



Arithmetic and Logic Unit – ALU

The arithmetic-logic unit or ALU is a critical area of the microcontroller that carries out arithmetic and logic operations of the instruction set. Connected to the main microcontroller data bus, the ALU receives related instruction codes and performs the required arithmetic or logical operations after which the result will be placed in the specified register. As these ALU calculation or operations may result in carry, borrow or other status changes, the status register will be correspondingly updated to reflect these changes. The ALU supports the following functions:

- Arithmetic operations:

  ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA,

  LADD, LADDM, LADC, LADCM, LSUB, LSUBM, LSBC, LSBCM, LDAA

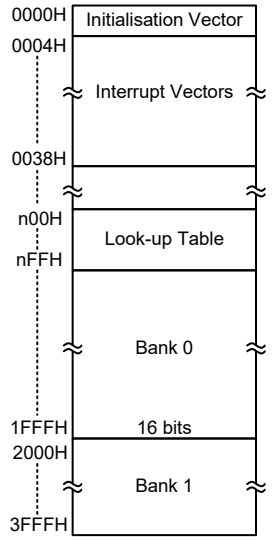
- Logic operations:
  AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA,
  LAND, LOR, LXOR, LANDM, LORM, LXORM, LCPL, LCPLA
- Rotation:
  RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC,
  LRRA, LRR, LRRCA, LRRC, LRLA, LRL, LRLCA, LRLC
- Increment and Decrement:
  INCA, INC, DECA, DEC,
  LINCA, LINC, LDECA, LDEC
- Branch decision:
  JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET, RETI,
  LSZ, LSZA, LSNZ, LSIZ, LSDZ, LSIZA, LSDZA

Flash Program Memory

The Program Memory is the location where the user code or program is stored. For the device the Program Memory is Flash type, which means it can be programmed and re-programmed a large number of times, allowing the user the convenience of code modification on the same device. By using the appropriate programming tools, the Flash device offers users the flexibility to conveniently debug and develop their applications while also offering a means of field programming and updating.

Structure

The Program Memory has a capacity of 16K×16 bits. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupt entries. Table data, which can be setup in any location within the Program Memory, is addressed by a separate table pointer registers.



Program Memory Structure

Special Vectors

Within the Program Memory, certain locations are reserved for the reset and interrupts. The location 000H is reserved for use by the device reset for program initialisation. After a device reset is initiated, the program will jump to this location and begin execution.

Look-up Table

Any location within the Program Memory can be defined as a look-up table where programmers can store fixed data. To use the look-up table, the table pointer must first be setup by placing the address of the look up data to be retrieved in the table pointer register, TBLP and TBHP. These registers define the total address of the look-up table.

After setting up the table pointer, the table data can be retrieved from the Program Memory using the "TABRD [m]" or "TABRDL [m]" instructions respectively when the memory [m] is located in sector 0. If the memory [m] is located in other sectors except sector 0, the data can be retrieved from the program memory using the corresponding extended table read instruction such as "LTABRD [m]" or "LTABRDL [m]" respectively. When the instruction is executed, the lower order table byte from the Program Memory will be transferred to the user defined Data Memory register [m] as specified in the instruction. The higher order table data byte from the Program Memory will be transferred to the TBLH special register. Any unused bits in this transferred higher order byte will be read as "0".

The accompanying diagram illustrates the addressing data flow of the look-up table.

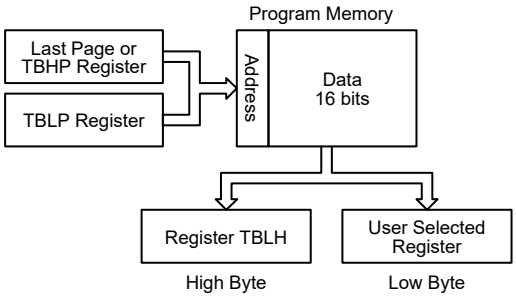


Table Program Example

The accompanying example shows how the table pointer and table data is defined and retrieved from the device. This example uses raw table data located in the last page which is stored there using the ORG statement. The value at this ORG statement is "1F00H" which is located in ROM Bank 1 and refers to the start address of the last page within the 16K words Program Memory. The table pointer low byte register is setup here to have an initial value of "06H". This will ensure that the first data read from the data table will be at the Program Memory address "3F06H" or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the specific address pointed by TBHP and TBLP if the "TABRD [m]" or "LTABRD [m]" instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the "TABRD [m]" or "LTABRD [m]" instruction is executed.

Because the TBLH register is a read/write register and can be restored, care should be taken to ensure its protection if both the main routine and Interrupt Service Routine use table read instructions. If using the table read instructions, the Interrupt Service Routines may change the value of the TBLH and subsequently cause errors if used again by the main routine. As a rule it is recommended that simultaneous use of the table read instructions should be avoided. However, in situations where simultaneous use cannot be avoided, the interrupts should be disabled prior to the execution of any main routine table-read instructions. Note that all table related instructions require two instruction cycles to complete their operation.

Table Read Program Example

```
rombank 1 code1
ds .section 'data'
tempreg1 db ?    ; temporary register #1
tempreg2 db ?    ; temporary register #2
```

```
code0 .section 'code'
mov a,06h          ; initialise table pointer - note that this address is referenced
mov tblp,a         ; to the last page or the page that tbhp pointed
mov a,3fh          ; initialise high table pointer
mov tbhp,a         ; it is not necessary to set tbhp if executing tabrdl or ltabrdl
:
:
tabrd tempreg1     ; transfers value in table referenced by table pointer data at
program
                   ; memory address "3F06H" transferred to tempreg1 and TBLH
dec tblp           ; reduce value of table pointer by one
tabrd tempreg2     ; transfers value in table referenced by table pointer data at
program
                   ; memory address "3F05H" transferred to tempreg2 and TBLH
                   ; in this example the data "1AH" is transferred to tempreg1 and
                   ; data "0FH" to tempreg2 the value "00H" will be
                   ; transferred to the high byte register TBLH
:
:
code1 .section 'code'
org 1F00h          ; sets initial address of last page
dc 00Ah,00Bh,00Ch,00Dh,00Eh,00Fh,01Ah,01Bh
```

In Circuit Programming – ICP

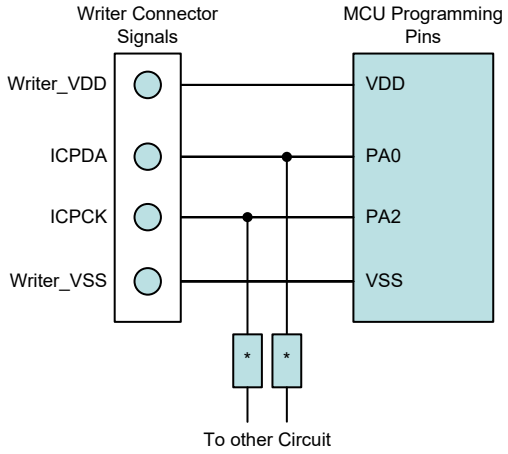
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device.

As an additional convenience, Holtek has provided a means of programming the microcontroller in-circuit using a 4-pin interface. This provides manufacturers with the possibility of manufacturing their circuit boards complete with a programmed or un-programmed microcontroller, and then programming or upgrading the program at a later stage. This enables product manufacturers to easily keep their manufactured products supplied with the latest program releases without removal and re-insertion of the device.

Holtek Writer Pins	MCU Programming Pins	Pin Description
ICPDA	PA0	Programming Serial Data/Address
ICPCK	PA2	Programming Clock
VDD	VDD	Power Supply
VSS	VSS	Ground

The Program Memory can be programmed serially in-circuit using this 4-wire interface. Data is downloaded and uploaded serially on a single pin with an additional line for the clock. Two additional lines are required for the power supply. The technical details regarding the in-circuit programming of the device are beyond the scope of this document and will be supplied in supplementary literature.

During the programming process, the user must take care of the ICPDA and ICPCK pins for data and clock programming purposes to ensure that no other outputs are connected to these two pins.



Note: * may be resistor or capacitor. The resistance of * must be greater than 1kΩ or the capacitance of * must be less than 1nF.

On-Chip Debug Support – OCDS

The device also provides the "On-Chip Debug" function to debug the MCU during development process. Users can use the OCDS function to emulate the device behaviors by connecting the OCDSDA and OCDSCK pins to the Holtek HT-IDE development tools. The OCDSDA pin is the OCDS Data/Address input/output pin while the OCDSCK pin is the OCDS clock input pin. When users use the OCDS function for debugging, the corresponding pin functions shared with the OCDSDA and OCDSCK pins in the device will have no effect. However, the two OCDS pins which are pin-shared with the ICP programming pins are still used as the Flash Memory programming pins for ICP. For more detailed OCDS information, refer to the corresponding document named "Holtek e-Link for 8-bit MCU OCDS User's Guide".

Holtek e-Link Pins	MCU OCDS Pins	Pin Description
OCDSDA	OCDSDA	On-Chip Debug Support Data/Address input/output
OCDSCK	OCDSCK	On-Chip Debug Support Clock input
VDD	VDD	Power Supply
VSS	VSS	Ground

In Application Programming – IAP

Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device. The provision of the IAP function offers users the convenience of Flash Memory multi-programming features. The convenience of the IAP function is that it can execute the updated program procedure using its internal firmware, without requiring an external Program Writer or PC. In addition, the IAP interface can also be any type of communication protocol, such as UART or USB, using I/O pins. Regarding the internal firmware, the user can select versions provided by Holtek or create their own. The following section illustrates the procedures regarding how to implement the IAP firmware.

Flash Memory Read/Write Size

The Flash memory Erase and Write operations are carried out in a page format while the Read operation is carried out in a word format. The page size and write buffer size are both assigned with a capacity of 32 words. Note that the Erase operation should be executed before the Write operation is executed.

When the Flash Memory Erase/Write Function is successfully enabled, the CFWEN bit will be set high. When the CFWEN bit is set high, the data can be written into the write buffer. The FWT bit is used to initiate the write process and then indicate the write operation status. This bit is set high by application programs to initiate a write process and will be cleared by hardware if the write process is finished.

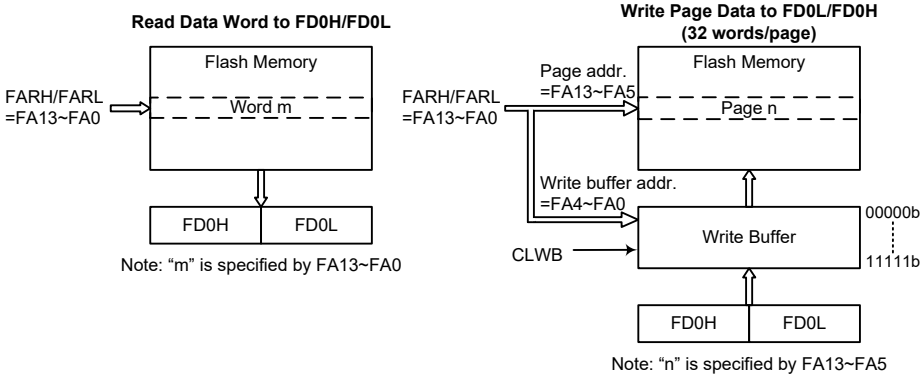
The Read operation can be carried out by executing a specific read procedure. The FRDEN bit is used to enable the read function and the FRD bit is used to initiate the read process by application programs and then indicate the read operation status. When the read process is finished, this bit will be cleared by hardware.

Operations	Format
Erase	32 words/page
Write	32 words/time
Read	1 word/time
Note: Page size = Write buffer size = 32 words.	

IAP Operation Format

Page	FARH	FARL[7:5]	FARL[4:0]
0	0000 0000	000	Tag Address
1	0000 0000	001	
2	0000 0000	010	
3	0000 0000	011	
4	0000 0000	100	
:	:	:	
510	0011 1111	110	
511	0011 1111	111	

Page Number and Address Selection



Flash Memory IAP Read/Write Structure

Write Buffer

The write buffer is used to store the written data temporarily when executing the write operation. The Write Buffer can be filled with written data after the Flash Memory Erase/Write Function has been successfully enabled by executing the Flash Memory Erase/Write Function Enable procedure. The write buffer can be cleared by configuring the CLWB bit in the FC2 register. The CLWB bit can be set high to enable the Clear Write Buffer procedure. When the procedure is finished this bit will be cleared to zero by hardware. It is recommended that the write buffer should be cleared by setting the CLWB bit high before the write buffer is used for the first time or when the data in the write buffer is updated.

The write buffer size is 32 words corresponding to a page. The write buffer address is mapped to a specific Flash memory page specified by the memory address bits, FA13~FA5. The data written into the FD0L and FD0H registers will be loaded into the write buffer. When data is written into the high byte data register, FD0H, it will result in the data stored in the high and low byte data registers both being written into the write buffer. It will also cause the Flash memory address to be incremented by one, after which the new address will be loaded into the FARH and FARL address registers. When the Flash memory address reaches the page boundary, 11111b of a page with 32 words, the address will now not be incremented but stop at the last address of the page. At this point a new page address should be specified for any other erase/write operations.

After a write process is finished, the write buffer will automatically be cleared by hardware. Note that the write buffer should be cleared manually by the application program when the data written into the Flash memory is incorrect in the data verification step. The data should again be written into the write buffer after the write buffer has been cleared when the data is found to be incorrect during the data verification step.

IAP Flash Program Memory Registers

There are two address registers, four 16-bit data registers and three control registers, which are all located in Sector 0. Read and Write operations to the Flash memory are carried out using 16-bit data operations using the address and data registers and the control register. Several registers control the overall operation of the internal Flash Program Memory. The address registers are named FARL and FARH, the data registers are named FDnL and FDnH and the control registers are named FC0, FC1 and FC2. As these registers are all located in Sector 0, they can be directly accessed in the same way as any other Special Function Register.

Register Name	Bit							
	7	6	5	4	3	2	1	0
FC0	CFWEN	FMOD2	FMOD1	FMOD0	FWPEN	FWT	FRDEN	FRD
FC1	D7	D6	D5	D4	D3	D2	D1	D0
FC2	—	—	—	—	—	—	FWERTS	CLWB
FARL	FA7	FA6	FA5	FA4	FA3	FA2	FA1	FA0
FARH	—	—	FA13	FA12	FA11	FA10	FA9	FA8
FD0L	D7	D6	D5	D4	D3	D2	D1	D0
FD0H	D15	D14	D13	D12	D11	D10	D9	D8
FD1L	D7	D6	D5	D4	D3	D2	D1	D0
FD1H	D15	D14	D13	D12	D11	D10	D9	D8
FD2L	D7	D6	D5	D4	D3	D2	D1	D0
FD2H	D15	D14	D13	D12	D11	D10	D9	D8
FD3L	D7	D6	D5	D4	D3	D2	D1	D0
FD3H	D15	D14	D13	D12	D11	D10	D9	D8

IAP Register List

- **FC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	CFWEN	FMOD2	FMOD1	FMOD0	FWPEN	FWT	FRDEN	FRD
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **CFWEN:** Flash Memory Erase/Write function enable control
0: Flash memory erase/write function is disabled
1: Flash memory erase/write function has been successfully enabled
When this bit is cleared to 0 by application program, the Flash memory erase/write function is disabled. Note that this bit cannot be set high by application programs.

Writing a "1" into this bit results in no action. This bit is used to indicate the Flash memory erase/write function status. When this bit is set to 1 by hardware, it means that the Flash memory erase/write function is enabled successfully. Otherwise, the Flash memory erase/write function is disabled if the bit is zero.

Bit 6~4 **FMOD2~FMOD0:** Flash memory Mode selection
000: Write Mode
001: Page Erase Mode
010: Reserved
011: Read Mode
100: Reserved
101: Reserved
110: Flash memory Erase/Write function Enable Mode
111: Reserved

These bits are used to select the Flash Memory operation modes. Note that the "Flash memory Erase/Write function Enable Mode" should first be successfully enabled before the Erase or Write Flash memory operation is executed.

Bit 3 **FWPEN:** Flash memory Erase/Write function enable procedure Trigger
0: Erase/Write function enable procedure is not triggered or procedure timer times out
1: Erase/Write function enable procedure is triggered and procedure timer starts to count

This bit is used to activate the Flash memory Erase/Write function enable procedure and an internal timer. It is set by the application programs and then cleared by hardware when the internal timer times out. The correct patterns must be written into the FD1L/FD1H, FD2L/FD2H and FD3L/FD3H register pairs respectively as soon as possible after the FWPEN bit is set high.

Bit 2 **FWT:** Flash memory write initiate control
0: Do not initiate Flash memory write or indicating that a Flash memory write process has completed
1: Initiate Flash memory write process

This bit is set by software and cleared by hardware when the Flash memory write process has completed.

Bit 1 **FRDEN:** Flash memory read enable control
0: Flash memory read disable
1: Flash memory read enable

This is the Flash memory Read Enable Bit which must be set high before any Flash memory read operations are carried out. Clearing this bit to zero will inhibit Flash memory read operations.

Bit 0 **FRD:** Flash memory read initiate control
0: Do not initiate Flash memory read or indicating that a Flash memory read process has completed
1: Initiate Flash memory read process

This bit is set by software and cleared by hardware when the Flash memory read process has completed.

Note: 1. The FWT, FRDEN and FRD bits cannot be set to "1" at the same time with a single instruction.
2. Ensure that the f_{SUB} clock is stable before executing the erase or write operation.
3. Note that the CPU will be stopped when a read, write or erase operation is successfully activated.
4. Ensure that the read, erase or write operation is totally complete before executing other operations.

• **FC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0:** Chip Reset Pattern

When a specific value of "55H" is written into this register, a reset signal will be generated to reset the whole chip.

• **FC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	FWERTS	CLWB
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1 **FWERTS:** Erase time and Write time selection
0: Erase time is 3.2ms (t_{FER}) / Write time is 2.2ms (t_{FWR})
1: Erase time is 3.7ms (t_{FER}) / Write time is 3.0ms (t_{FWR})

Bit 0 **CLWB:** Flash memory Write Buffer Clear control
0: Do not initiate a Write Buffer Clear process or indicating that a Write Buffer Clear process has completed
1: Initiate Write Buffer Clear process
This bit is set by software and cleared by hardware when the Write Buffer Clear process has completed.

• **FARL Register**

Bit	7	6	5	4	3	2	1	0
Name	FA7	FA6	FA5	FA4	FA3	FA2	FA1	FA0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **FA7~FA0:** Flash Memory Address bit 7 ~ bit 0

• **FARH Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	FA13	FA12	FA11	FA10	FA9	FA8
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5~0 **FA13~FA8:** Flash Memory Address bit 13 ~ bit 8

• **FD0L Register**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0:** The first Flash Memory data word bit 7 ~ bit 0
Note that data written into the low byte data register FD0L will only be stored in the FD0L register and not loaded into the lower 8-bit write buffer.

• **FD0H Register**

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D15~D8:** The first Flash Memory data word bit 15 ~ bit 8
Note that when 8-bit data is written into the high byte data register FD0H, the whole 16 bits of data stored in the FD0H and FD0L registers will simultaneously be loaded into the 16-bit write buffer after which the contents of the Flash memory address register pair, FARH and FARL, will be incremented by one.

• **FD1L Register**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0:** The second Flash Memory data word bit 7 ~ bit 0

• **FD1H Register**

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D15~D8:** The second Flash Memory data word bit 15 ~ bit 8

• **FD2L Register**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0:** The third Flash Memory data word bit 7 ~ bit 0

• **FD2H Register**

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D15~D8:** The third Flash Memory data word bit 15 ~ bit 8

• **FD3L Register**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0:** The fourth Flash Memory data word bit 7 ~ bit 0

• **FD3H Register**

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D15~D8:** The fourth Flash Memory data word bit 15 ~ bit 8

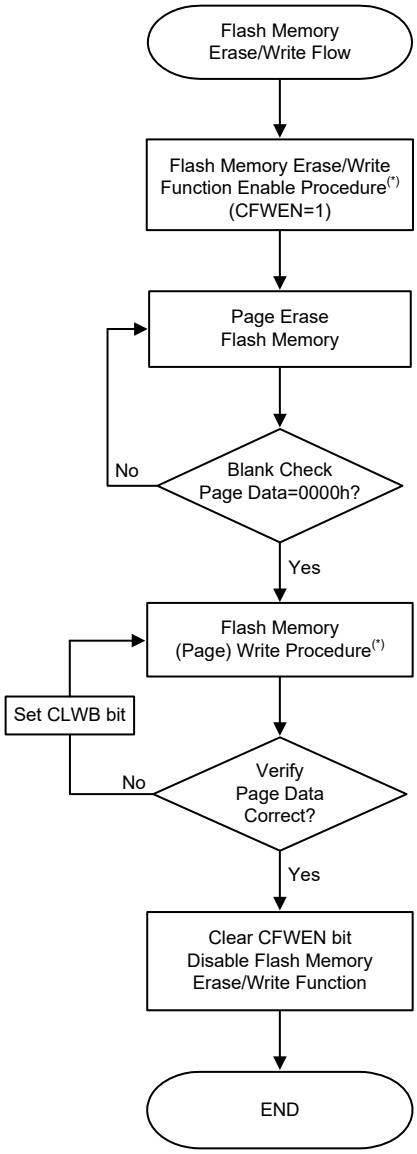
Flash Memory Erase/Write Flow

It is important to understand the Flash memory Erase/Write flow before the Flash memory contents are updated. Users can refer to the corresponding operation procedures when developing their IAP program to ensure that the Flash memory contents are correctly updated.

Flash Memory Erase/Write Flow Descriptions

1. Activate the "Flash Memory Erase/Write function enable procedure" first. When the Flash Memory Erase/Write function is successfully enabled, the CFWEN bit in the FC0 register will automatically be set high by hardware. After this, Erase or Write operations can be executed on the Flash memory. Refer to the "Flash Memory Erase/Write Function Enable Procedure" for details.
2. Configure the Flash memory address to select the desired erase page, tag address and then erase this page.

   For a page erase operation, set the FARL and FARH registers to specify the start address of the erase page, then write dummy data into the FD0H register to tag address. The current address will be internally incremented by one after each dummy data is written into the FD0H register. When the address reaches the page boundary, 11111b, the address will not be further incremented but stop at the last address of the page. Note that the write operation to the FD0H register is used to tag address, it must be implemented to determine which addresses to be erased.
3. Execute a Blank Check operation to ensure whether the page erase operation is successful or not. The "TABRD" instruction should be executed to read the Flash memory contents and to check if the contents is 0000h or not. If the Flash memory page erase operation fails, users should go back to Step 2 and execute the page erase operation again.
4. Write data into the specific page. Refer to the "Flash Memory Write Procedure" for details.
5. Execute the "TABRD" instruction to read the Flash memory contents and check if the written data is correct or not. If the data read from the Flash memory is different from the written data, it means that the page write operation has failed. The CLWB bit should be set high to clear the write buffer and then write the data into the specific page again if the write operation has failed.
6. Clear the CFWEN bit to disable the Flash Memory Erase/Write function enable mode if the current page Erase and Write operations are complete if no more pages need to be erased or written.



Flash Memory Erase/Write Flow

Note: The Flash Memory Erase/Write Function Enable procedure and Flash Memory Write procedure will be described in the following sections.

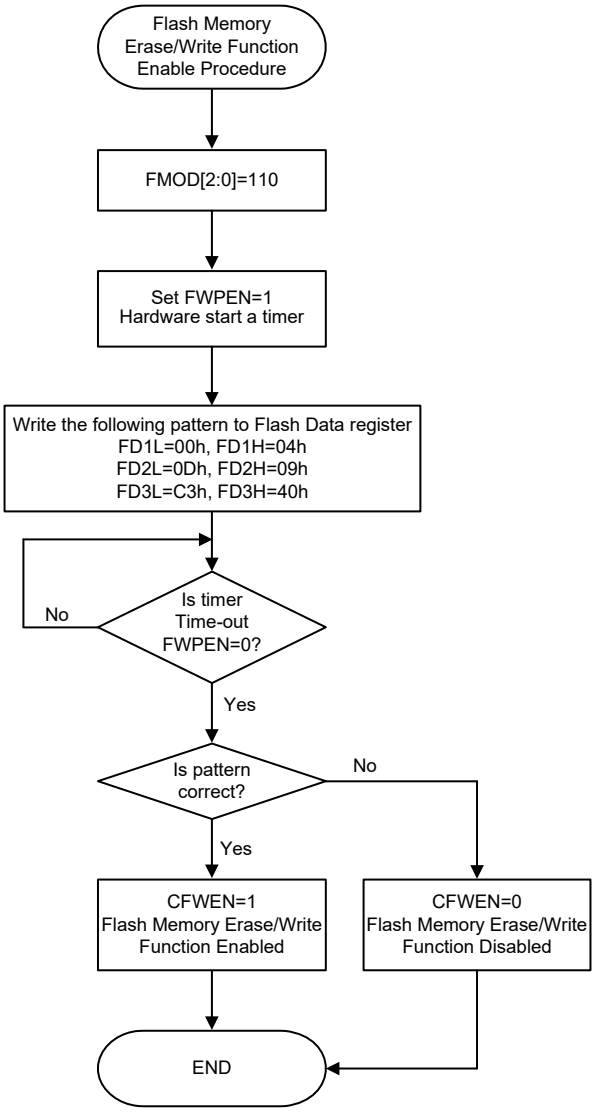
Flash Memory Erase/Write Function Enable Procedure

The Flash Memory Erase/Write Function Enable Mode is specially designed to prevent the Flash memory contents from being wrongly modified. In order to allow users to change the Flash memory data using the IAP control registers, users must first enable the Flash memory Erase/Write function.

Flash Memory Erase/Write Function Enable Procedure Description

1. Write data "110" to the FMOD[2:0] bits in the FC0 register to select the Flash Memory Erase/Write Function Enable Mode.
2. Set the FWPEN bit in the FC0 register to "1" to activate the Flash Memory Erase/Write Function. This will also activate an internal timer.
3. Write the correct data pattern into the Flash data registers, FD1L~FD3L and FD1H~FD3H, as soon as possible after the FWPEN bit is set high. The enable Flash memory erase/write function data pattern is 00H, 0DH, C3H, 04H, 09H and 40H corresponding to the FD1L~FD3L and FD1H~FD3H registers respectively.
4. Once the timer has timed out, the FWPEN bit will automatically be cleared to 0 by hardware regardless of the input data pattern.
5. If the written data pattern is incorrect, the Flash memory erase/write function will not be enabled successfully and the above steps should be repeated. If the written data pattern is correct, the Flash memory erase/write function will be enabled successfully.
6. Once the Flash memory erase/write function is enabled, the Flash memory contents can be updated by executing the page erase and write operations using the IAP control registers.

To disable the Flash memory erase/write function, the CFWEN bit in the FC0 register can be cleared. There is no need to execute the above procedure.



Flash Memory Erase/Write Function Enable Procedure

Flash Memory Write Procedure

After the Flash memory erase/write function has been successfully enabled as the CFWEN bit is set high, the data to be written into the Flash memory can be loaded into the write buffer. The selected Flash memory page data should be erased by properly configuring the IAP control registers before the data write procedure is executed.

The write buffer size is 32 words, known as a page, whose address is mapped to a specific Flash memory page specified by the memory address bits, FA13~FA5. It is important to ensure that the page where the write buffer data is located is the same one which the memory address bits specify.

Flash Memory Consecutive Write Description

The maximum amount of write data is 32 words for each write operation. The write buffer address will be automatically incremented by one when consecutive write operations are executed. The start address of a specific page should first be written into the FARL and FARH registers. Then the data word should be written into the FD0L register and then the FD0H register. At the same time the write buffer address will be incremented by one and then the next data word can be written into the FD0L and FD0H registers for the next address without modifying the address register pair, FARH and FARL. When the write buffer address reaches the page boundary the address will not be further incremented but will stop at the last address of the page.

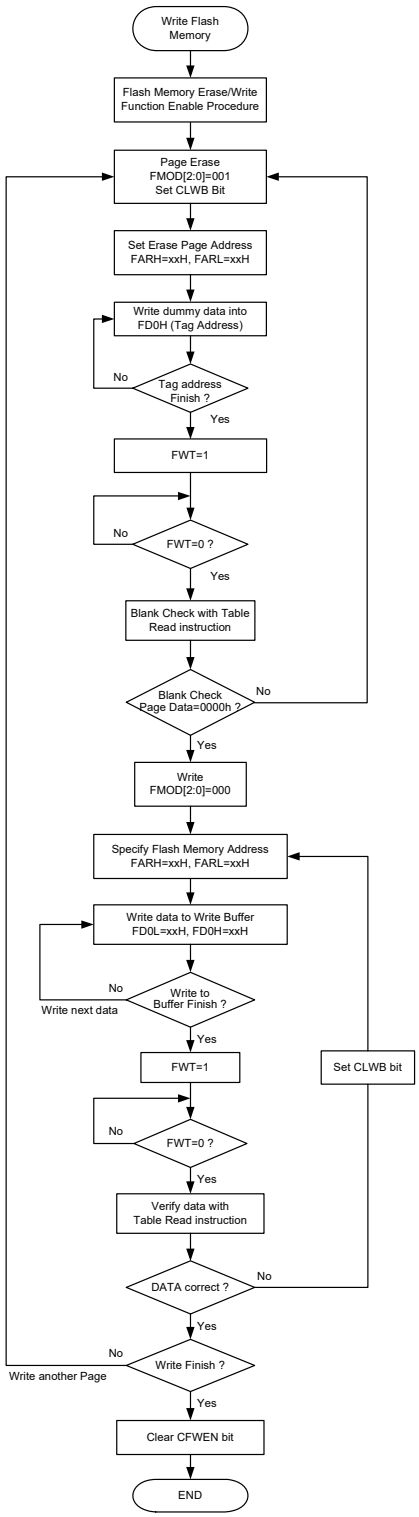
1. Activate the "Flash Memory Erase/Write function enable procedure". Check the CFWEN bit value and then execute the erase/write operations if the CFWEN bit is set high. Refer to the "Flash Memory Erase/Write function enable procedure" for more details.
2. Set the FMOD field to "001" to select the erase operation and set the CLWB bit high to clear the write buffer. Set the FWT bit high to erase the desired page which is specified by the FARH and FARL registers and has been tagged address. Wait until the FWT bit goes low.
3. Execute a Blank Check operation using the table read instruction to ensure that the erase operation has successfully completed.

   Go to step 2 if the erase operation is not successful.

   Go to step 4 if the erase operation is successful.
4. Set the FMOD field to "000" to select the write operation.
5. Setup the desired start address in the FARH and FARL registers. Write the desired data words consecutively into the FD0L and FD0H registers within a page as specified by their consecutive addresses. The maximum written data number is 32 words.
6. Set the FWT bit high to write the data words from the write buffer to the Flash memory. Wait until the FWT bit goes low.
7. Verify the data using the table read instruction to ensure that the write operation has successfully completed.

   If the write operation has not successfully completed, set the CLWB bit high to clear the write buffer and then go to step 5.

   Go to step 8 if the write operation is successful.
8. Clear the CFWEN bit low to disable the Flash memory erase/write function.



Flash Memory Consecutive Write Procedure

Note: 1. When the erase or write operation is successfully activated, all CPU operations will temporarily cease.

2. It will take certain time for the FWT bit state changing from high to low in the erase or write operation, which can be selected by the FWERTS bit in the FC2 register.

Flash Memory Non-consecutive Write Description

The main difference between Flash Memory Consecutive and Non-consecutive Write operations is whether the data words to be written are located in consecutive addresses or not. If the data to be written is not located in consecutive addresses the desired address should be re-assigned after a data word is successfully written into the Flash Memory.

A two data word non-consecutive write operation is taken as an example here and described as follows:

1. Activate the "Flash Memory Erase/Write function enable procedure". Check the CFWEN bit value and then execute the erase/write operation if the CFWEN bit is set high. Refer to the "Flash Memory Erase/Write function enable procedure" for more details.
2. Set the FMOD field to "001" to select the erase operation and set the CLWB bit high to clear the write buffer. Set the FWT bit high to erase the desired page which is specified by the FARH and FARL registers and has been tagged address. Wait until the FWT bit goes low.
3. Execute a Blank Check operation using the table read instruction to ensure that the erase operation has successfully completed.

   Go to step 2 if the erase operation is not successful.

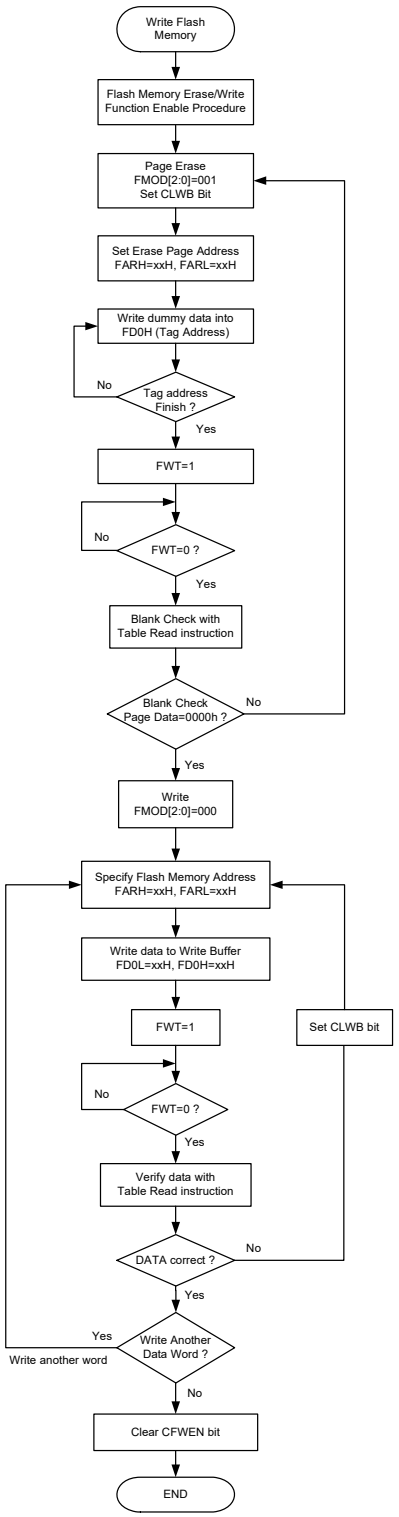
   Go to step 4 if the erase operation is successful.
4. Set the FMOD field to "000" to select the write operation.
5. Setup the desired address ADDR1 in the FARH and FRARL registers. Write the desired data word DATA1 first into the FD0L register and then into the FD0H register.
6. Set the FWT bit high to transfer the data word from the write buffer to the Flash memory. Wait until the FWT bit goes low.
7. Verify the data using the table read instruction to ensure that the write operation has successfully completed.

   If the write operation has not successfully completed, set the CLWB bit high to clear the write buffer and then go to step 5.

   Go to step 8 if the write operation is successful.
8. Setup the desired address ADDR2 in the FARH and FRARL registers. Write the desired data word DATA2 first into the FD0L register and then into the FD0H register.
9. Set the FWT bit high to transfer the data word from the write buffer to the Flash memory. Wait until the FWT bit goes low.
10. Verify the data using the table read instruction to ensure that the write operation has successfully completed.

    If the write operation has not successfully completed, set the CLWB bit high to clear the write buffer and then go to step 8.

    Go to step 11 if the write operation is successful.
11. Clear the CFWEN bit low to disable the Flash memory erase/write function.



Flash Memory Non-consecutive Write Procedure

Note: 1. When the erase or write operation is successfully activated, all CPU operations will temporarily cease.

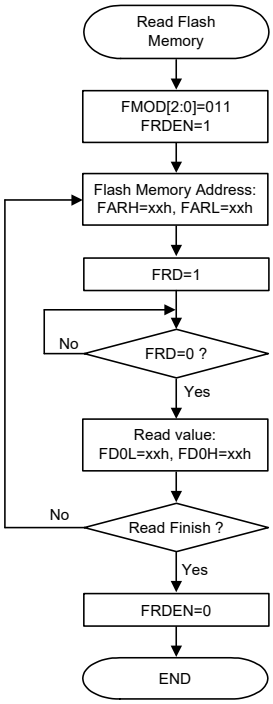
2. It will take certain time for the FWT bit state changing from high to low in the erase or write operation, which can be selected by the FWERTS bit in the FC2 register.

Important Points to Note for Flash Memory Write Operations

1. The "Flash Memory Erase/Write Function Enable Procedure" must be successfully activated before the Flash Memory erase/write operation is executed.
2. The Flash Memory erase operation is executed to erase a whole page.
3. The whole write buffer data will be written into the Flash memory in a page format. The corresponding address cannot exceed the page boundary.
4. After the data is written into the Flash memory the Flash memory contents must be read out using the table read instruction, TABRD, and checked if it is correct or not. If the data written into the Flash memory is incorrect, the write buffer should be cleared by setting the CLWB bit high and then writing the data again into the write buffer. Then activate a write operation on the same Flash memory page without erasing it. The data check, buffer clear and data re-write steps should be repeatedly executed until the data written into the Flash memory is correct.
5. The system frequency should be setup to the maximum application frequency when data write and data check operations are executed using the IAP function.

Flash Memory Read Procedure

To activate the Flash Memory Read procedure, the FMOD field should be set to "011" to select the Flash memory read mode and the FRDEN bit should be set high to enable the read function. The desired Flash memory address should be written into the FARH and FARL registers and then the FRD bit should be set high. After this the Flash memory read operation will be activated. The data stored in the specified address can be read from the data registers, FD0H and FD0L, when the FRD bit goes low. There is no need to first activate the Flash Memory Erase/Write Function Enable Procedure before the Flash memory read operation is executed.



Flash Memory Read Procedure

Note: 1. When the read operation is successfully activated, all CPU operations will temporarily cease.
2. It will take a typical time of three instruction cycles for the FRD bit state changing from high to low.

Data Memory

The Data Memory is an 8-bit wide RAM internal memory and is the location where temporary information is stored.

Divided into two types, the first of Data Memory is an area of RAM where special function registers are located. These registers have fixed locations and are necessary for correct operation of the device. Many of these registers can be read from and written to directly under program control, however, some remain protected from user manipulation. The second area of Data Memory is reserved for general purpose use. All locations within this area are read and write accessible under program control.

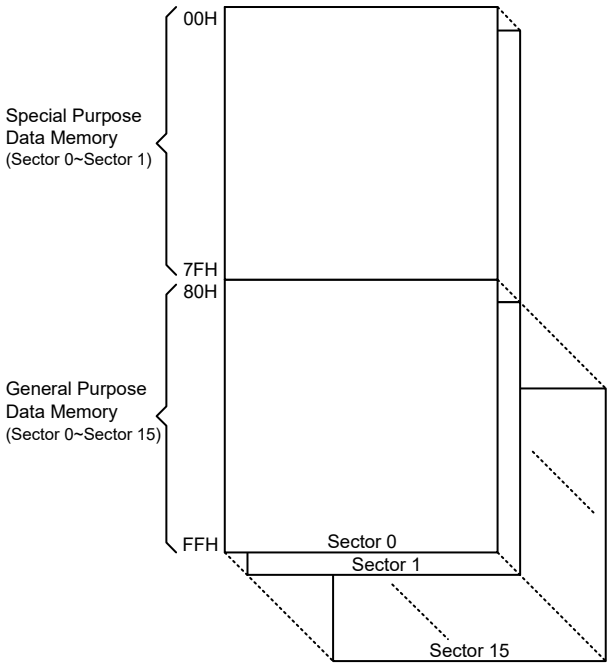
Structure

The Data Memory is subdivided into several sectors, all of which are implemented in 8-bit wide Memory. Each of the Data Memory sectors is generally categorized into two types, the Special Purpose Data Memory and the General Purpose Data Memory.

The address range of the Special Purpose Data Memory for the device is from 00H to 7FH while the General Purpose Data Memory address range is from 80H to FFH. Switching between the different Data Memory sectors is achieved by properly setting the Memory Pointers to correct value if using the indirect addressing method.

Special Purpose Data Memory	General Purpose Data Memory	
Located Sectors	Capacity	Sector: Address
0, 1	2048×8	0: 80H~FFH 1: 80H~FFH : 15: 80H~FFH

Data Memory Summary



Data Memory Structure

Data Memory Addressing

For the device that supports the extended instructions, there is no Bank Pointer for Data Memory. The Bank Pointer, PBP, is only available for Program Memory. For Data Memory the desired Sector is pointed by the MP1H or MP2H register and the certain Data Memory address in the selected sector is specified by the MP1L or MP2L register when using indirect addressing access.

Direct Addressing can be used in all sectors using the extended instruction which can address all available data memory space. For the accessed data memory which is located in any data memory sectors except sector 0, the extended instructions can be used to access the data memory instead of using the indirect addressing access. The main difference between standard instructions and extended instructions is that the data memory address "m" in the extended instructions has up to 12 valid bits for the device, the high byte indicates a sector and the low byte indicates a specific address.

General Purpose Data Memory

All microcontroller programs require an area of read/write memory where temporary data can be stored and retrieved for use later. It is this area of RAM memory that is known as General Purpose Data Memory. This area of Data Memory is fully accessible by the user programming for both reading and writing operations. By using the bit operation instructions individual bits can be set or reset under program control giving the user a large range of flexibility for bit manipulation in the Data Memory.

Special Purpose Data Memory

This area of Data Memory is where registers, necessary for the correct operation of the microcontroller, are stored. Most of the registers are both readable and writeable but some are protected and are readable only, the details of which are located under the relevant Special Function Register section. Note that for locations that are unused, any read instruction to these addresses will return the value "00H".

	Sector 0	Sector 1
00H	IAR0	PTM0C0
01H	MP0	PTM0C1
02H	IAR1	PTM0DL
03H	MP1L	PTM0DH
04H	MP1H	PTM0AL
05H	ACC	PTM0AH
06H	PCL	PTM0RPL
07H	TBLP	PTM0RPH
08H	TBLH	PTM1C0
09H	TBHP	PTM1C1
0AH	STATUS	PTM1DL
0BH	PBP	PTM1DH
0CH	IAR2	PTM1AL
0DH	MP2L	PTM1AH
0EH	MP2H	PTM1RPL
0FH	RSTFC	PTM1RPH
10H	LVRC	PTM2C0
11H	LVDC	PTM2C1
12H	SCC	PTM2DL
13H	HIRCC	PTM2DH
14H	PA	PTM2AL
15H	PAC	PTM2AH
16H	PAPU	PTM2RPL
17H	PAWU	PTM2RPH
18H	PB	PTM3C0
19H	PBC	PTM3C1
1AH	PBPU	PTM3DL
1BH	PC	PTM3DH
1CH	PCC	PTM3AL
1DH	PCPU	PTM3AH
1EH		PTM3RPL
1FH	PDC	PTM3RPH
20H	PDPUS0	
21H	PAS0	
22H	PAS1	
23H	PBS0	
24H	PBS1	
25H	PCS0	
26H	PCS1	
27H	PDS0	
28H	IECC	
29H	WDTC	
2AH	TB0C	
2BH	TB1C	
2CH	SADOL	
2DH	SADOH	
2EH	SADC0	
2FH	SADC1	
30H	SADC2	
31H	VBGRC	
32H	INTC0	
33H	INTC1	
34H	INTC2	
35H	INTC3	
36H	MFI0	
37H	MFI1	
38H	MFI2	
39H	INTEG	
3AH	PSC0R	
3BH	PSC1R	
3CH	PMPS	
3DH	SLEDC0	
3EH	SLEDC1	
3FH	ADUDA0	

	Sector 0	Sector 1
40H	ADUDA1	EEC
41H	ADUC0	
42H	ADUC1	
43H	FC0	
44H	FC1	
45H	FC2	
46H	FARL	
47H	FARH	
48H	FD0L	
49H	FD0H	
4AH	FD1L	
4BH	FD1H	
4CH	FD2L	
4DH	FD2H	
4EH	FD3L	
4FH	FD3H	
50H	IICMC0	
51H	IICMC1	
52H	IICMC2	
53H	IICMD	
54H	IICMA	
55H	IICMC	
56H	EEAL	
57H	EEAH	
58H	EED	
59H	MDUWR0	
5AH	MDUWR1	
5BH	MDUWR2	
5CH	MDUWR3	
5DH	MDUWR4	
5EH	MDUWR5	
5FH	MDUWCTRL	
60H	OPC0	
61H	OPC1	
62H	OPOCAL	
63H	CRCCR	
64H	CRCIN	
65H	CRCDL	
66H	CRCDH	
67H	SIMC0	
68H	SIMC1	
69H	SIMD	
6AH	SIMA/SIMC2	
6BH	SIMTOC	
6CH	USR	
6DH	UCR1	
6EH	UCR2	
6FH	UCR3	
70H	BRDH	
71H	BRDL	
72H	UFCR	
73H	TXR_RXR	
74H	RxCNT	
75H	STMC0	
76H	STMC1	
77H	STMDL	
78H	STMDH	
79H	STMAL	
7AH	STMAH	
7BH	STMRP	
7CH		
7DH	IFS	
7EH		
7FH		

: Unused, read as 00H

: Reserved, cannot be changed

Special Purpose Data Memory Structure

Special Function Register Description

Most of the Special Function Register details will be described in the relevant functional section. However, several registers require a separate description in this section.

Indirect Addressing Registers – IAR0, IAR1, IAR2

The Indirect Addressing Registers, IAR0, IAR1 and IAR2, although having their locations in normal RAM register space, do not actually physically exist as normal registers. The method of indirect addressing for RAM data manipulation uses these Indirect Addressing Registers and Memory Pointers, in contrast to direct memory addressing, where the actual memory address is specified. Actions on the IAR0, IAR1 and IAR2 registers will result in no actual read or write operation to these registers but rather to the memory location specified by their corresponding Memory Pointers, MP0, MP1L/MP1H or MP2L/MP2H. Acting as a pair, IAR0 and MP0 can together access data only from Sector 0 while the IAR1 register together with MP1L/MP1H register pair and IAR2 register together with MP2L/MP2H register pair can access data from any Data Memory sector. As the Indirect Addressing Registers are not physically implemented, reading the Indirect Addressing Registers indirectly will return a result of "00H" and writing to the registers indirectly will result in no operation.

Memory Pointers – MP0, MP1H/MP1L, MP2H/MP2L

Five Memory Pointers, known as MP0, MP1L, MP1H, MP2L and MP2H, are provided. These Memory Pointers are physically implemented in the Data Memory and can be manipulated in the same way as normal registers providing a convenient way with which to address and track data. When any operation to the relevant Indirect Addressing Registers is carried out, the actual address that the microcontroller is directed to is the address specified by the related Memory Pointer. MP0, together with Indirect Addressing Register, IAR0, are used to access data from Sector 0, while MP1L/MP1H together with IAR1 and MP2L/MP2H together with IAR2 are used to access data from all data sectors according to the corresponding MP1H or MP2H register. Direct Addressing can be used in all data sectors using the extended instruction which can address all available data memory space.

The following example shows how to clear a section of four Data Memory locations already defined as locations adres1 to adres4.

Indirect Addressing Program Example

Example 1

```
data .section 'data'
adres1 db ?
adres2 db ?
adres3 db ?
adres4 db ?
block  db ?
code .section at 0 code
org 00h
start:
mov a,04h               ; setup size of block
mov block,a
mov a,offset adres1     ; Accumulator loaded with first RAM address
mov mp0,a               ; setup memory pointer with first RAM address
loop:
clr IAR0                ; clear the data at address defined by MP0
inc mp0                 ; increment memory pointer
sdz block               ; check if last memory location has been cleared
jmp loop
continue:
```

Example 2

```
data .section 'data'
adres1 db ?
adres2 db ?
adres3 db ?
adres4 db ?
block  db ?
code .section at 0 'code'
org 00h
start:
mov a,04h               ; setup size of block
mov block,a
mov a,01h               ; setup the memory sector
mov mp1h,a
mov a,offset adres1     ; Accumulator loaded with first RAM address
mov mp1l,a              ; setup memory pointer with first RAM address
loop:
clr IAR1                ; clear the data at address defined by MP1L
inc mp1l                ; increment memory pointer MP1L
sdz block               ; check if last memory location has been cleared
jmp loop
continue:
:
```

The important point to note here is that in the example shown above, no reference is made to specific RAM addresses.

Direct Addressing Program Example using extended instructions

```
data .section 'data'
temp db ?
code .section at 0 code
org  00h
start:
lmov a,[m]              ; move [m] data to acc
lsub a, [m+1]           ; compare [m] and [m+1] data
snz  c                  ; [m]>[m+1]?
jmp  continue           ; no
lmov a,[m]              ; yes, exchange [m] and [m+1] data
mov  temp,a
lmov a,[m+1]
lmov [m],a
mov  a,temp
lmov [m+1],a
continue:
:
```

Note: Here “m” is a data memory address located in any data memory sectors. For example, m=1F0H, it indicates address 0F0H in Sector 1.

Program Memory Bank Pointer – PBP

For the device the Program Memory is divided into several banks. Selecting the required Program Memory area is achieved using the Program Memory Bank Pointer, PBP. The PBP register should be properly configured before the device executes the "Branch" operation using the "JMP" or "CALL" instruction. After that a jump to a non-consecutive Program Memory address which is located in a certain bank selected by the program memory bank pointer bits will occur.

• **PBP Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	—	PBP0
R/W	—	—	—	—	—	—	—	R/W
POR	—	—	—	—	—	—	—	0

Bit 7~1 Unimplemented, read as "0"

Bit 0 **PBP0:** Program Memory Bank Point bit 0
0: Bank 0
1: Bank 1

Accumulator – ACC

The Accumulator is central to the operation of any microcontroller and is closely related with operations carried out by the ALU. The Accumulator is the place where all intermediate results from the ALU are stored. Without the Accumulator it would be necessary to write the result of each calculation or logical operation such as addition, subtraction, shift, etc., to the Data Memory resulting in higher programming and timing overheads. Data transfer operations usually involve the temporary storage function of the Accumulator; for example, when transferring data between one user-defined register and another, it is necessary to do this by passing the data through the Accumulator as no direct transfer between two registers is permitted.

Program Counter Low Register – PCL

To provide additional program control functions, the low byte of the Program Counter is made accessible to programmers by locating it within the Special Purpose area of the Data Memory. By manipulating this register, direct jumps to other program locations are easily implemented. Loading a value directly into this PCL register will cause a jump to the specified Program Memory location; however, as the register is only 8-bit wide, only jumps within the current Program Memory page are permitted. When such operations are used, note that a dummy cycle will be inserted.

Look-up Table Registers – TBLP, TBHP, TBLH

These three special function registers are used to control operation of the look-up table which is stored in the Program Memory. The TBLP and TBHP registers are the table pointer pair and indicates the location where the table data is located. Their value must be setup before any table read instructions are executed. Their value can be changed, for example using the "INC" or "DEC" instructions, allowing for easy table data pointing and reading. TBLH is the location where the high order byte of the table data is stored after a table read data instruction has been executed. Note that the lower order table data byte is transferred to a user defined location.

Status Register – STATUS

This 8-bit register contains the zero flag (Z), carry flag (C), auxiliary carry flag (AC), overflow flag (OV), SC flag, CZ flag, power down flag (PDF), and watchdog time-out flag (TO). These arithmetic/logical operation and system management flags are used to record the status and operation of the microcontroller.

With the exception of the TO and PDF flags, bits in the status register can be altered by instructions like most other registers. Any data written into the status register will not change the TO or PDF flag. In addition, operations related to the status register may give different results due to the different instruction operations. The TO flag can be affected only by a system power-up, a WDT time-out or by executing the "CLR WDT" or "HALT" instruction. The PDF flag is affected only by executing the "HALT" or "CLR WDT" instruction or during a system power-up.

The Z, OV, AC, C, SC and CZ flags generally reflect the status of the latest operations.

- C is set if an operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation; otherwise C is cleared. C is also affected by a rotate through carry instruction.
- AC is set if an operation results in a carry out of the low nibbles in addition, or no borrow from the high nibble into the low nibble in subtraction; otherwise AC is cleared.
- Z is set if the result of an arithmetic or logical operation is zero; otherwise Z is cleared.
- OV is set if an operation results in a carry into the highest-order bit but not a carry out of the highest-order bit, or vice versa; otherwise OV is cleared.
- PDF is cleared by a system power-up or executing the "CLR WDT" instruction. PDF is set by executing the "HALT" instruction.
- TO is cleared by a system power-up or executing the "CLR WDT" or "HALT" instruction. TO is set by a WDT time-out.
- SC is the result of the "XOR" operation which is performed by the OV flag and the MSB of the current instruction operation result.
- CZ is the operational result of different flags for different instructions. Refer to register definitions for more details.

In addition, on entering an interrupt sequence or executing a subroutine call, the status register will not be pushed onto the stack automatically. If the contents of the status registers are important and if the subroutine can corrupt the status register, precautions must be taken to correctly save it.

• STATUS Register

Bit	7	6	5	4	3	2	1	0
Name	SC	CZ	TO	PDF	OV	Z	AC	C
R/W	R/W	R/W	R	R	R/W	R/W	R/W	R/W
POR	x	x	0	0	x	x	x	x

"x": unknown

Bit 7 **SC:** The result of the "XOR" operation which is performed by the OV flag and the MSB of the instruction operation result

Bit 6 **CZ:** The operational result of different flags for different instructions
For SUB/SUBM/LSUB/LSUBM instructions, the CZ flag is equal to the Z flag.
For SBC/ SBCM/ LSBC/ LSBCM instructions, the CZ flag is the "AND" operation result which is performed by the previous operation CZ flag and current operation zero flag. For other instructions, the CZ flag will not be affected.

Bit 5 **TO:** Watchdog Time-out flag
0: After power up or executing the "CLR WDT" or "HALT" instruction
1: A watchdog time-out occurred

Bit 4 **PDF:** Power down flag
0: After power up or executing the "CLR WDT" instruction
1: By executing the "HALT" instruction

Bit 3 **OV:** Overflow flag
0: No overflow
1: An operation results in a carry into the highest-order bit but not a carry out of the highest-order bit or vice versa

Bit 2 **Z:** Zero flag
0: The result of an arithmetic or logical operation is not zero
1: The result of an arithmetic or logical operation is zero

Bit 1 **AC:** Auxiliary flag
0: No auxiliary carry
1: An operation results in a carry out of the low nibbles, in addition, or no borrow from the high nibble into the low nibble in subtraction

Bit 0 **C:** Carry flag
0: No carry-out
1: An operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation

The "C" flag is also affected by a rotate through carry instruction.

EEPROM Data Memory

The device contains an area of internal EEPROM Data Memory. EEPROM is by its nature a non-volatile form of re-programmable memory, with data retention even when its power supply is removed. By incorporating this kind of data memory, a whole new host of application possibilities are made available to the designer. The availability of EEPROM storage allows information such as product identification numbers, calibration values, specific user data, system setup data or other product information to be stored directly within the product microcontroller. The process of reading and writing data to the EEPROM memory has been reduced to a very trivial affair.

EEPROM Data Memory Structure

The EEPROM Data Memory capacity is 1024×8 bits for the device. Unlike the Program Memory and RAM Data Memory, the EEPROM Data Memory is not directly mapped into memory space and is therefore not directly addressable in the same way as the other types of memory. Read and write operations to the EEPROM are carried out in either the byte mode or page mode determined by the mode selection bit, MODE, in the control register, EEC.

EEPROM Registers

Four registers control the overall operation of the internal EEPROM Data Memory. These are the address registers, EEAL and EEAH, the data register, EED and a single control register, EEC. As the EEAL, EEAH and EED registers are located in Sector 0, they can be directly accessed in the same way as any other Special Function Register. The EEC register, however, being located in Sector 1, can only be read from or written to indirectly using the MP1H/MP1L or MP2H/MP2L Memory Pointer pair and Indirect Addressing Register, IAR1 or IAR2. Because the EEC control register is located at address 40H in Sector 1, the Memory Pointer low byte register, MP1L or MP2L, must first be set to the value 40H and the Memory Pointer high byte register, MP1H or MP2H, set to the value, 01H, before any operations on the EEC register are executed.

Register Name	Bit							
	7	6	5	4	3	2	1	0
EEAL	EEAL7	EEAL6	EEAL5	EEAL4	EEAL3	EEAL2	EEAL1	EEAL0
EEAH	—	—	—	—	—	—	EEAH1	EEAH0
EED	D7	D6	D5	D4	D3	D2	D1	D0
EEC	EWERTS	EREN	ER	MODE	WREN	WR	RDEN	RD

EEPROM Register List

- **EEAL Register**

Bit	7	6	5	4	3	2	1	0
Name	EEAL7	EEAL6	EEAL5	EEAL4	EEAL3	EEAL2	EEAL1	EEAL0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **EEAL7~EEAL0:** Data EEPROM low byte address bit 7 ~ bit 0

- **EEAH Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	EEAH1	EEAH0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **EEAH1~EEAH0:** Data EEPROM high byte address bit 1 ~ bit 0

- **EED Register**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0:** Data EEPROM data bit 7 ~ bit 0

- **EEC Register**

Bit	7	6	5	4	3	2	1	0
Name	EWERTS	EREN	ER	MODE	WREN	WR	RDEN	RD
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **EWERTS:** Data EEPROM Erase time and Write time selection
0: Erase time is 3.2ms (t_{EEER}) / Write time is 2.2ms (t_{EEWR})
1: Erase time is 3.7ms (t_{EEER}) / Write time is 3.0ms (t_{EEWR})

Bit 6 **EREN:** Data EEPROM erase enable
0: Disable
1: Enable
This bit is used to enable Data EEPROM erase function and must be set high before Data EEPROM erase operations are carried out. This bit will be automatically reset to zero by hardware after the erase cycle has finished. Clearing this bit to zero will inhibit data EEPROM erase operations.

Bit 5 **ER:** Data EEPROM erase control
0: Erase cycle has finished
1: Activate an erase cycle
This is the Data EEPROM Erase Control Bit. When this bit is set high by the application program, an erase cycle will be activated. This bit will be automatically

reset to zero by hardware after the erase cycle has finished. Setting this bit high will have no effect if the EREN has not first been set high.

Bit 4 **MODE:** Data EEPROM operation mode selection
0: Byte operation mode
1: Page operation mode
This is the EEPROM operation mode selection bit. When the bit is set high by the application program, the Page write, erase or read function will be selected. Otherwise, the byte write or read function will be selected. The EEPROM page buffer size is 16 bytes.

Bit 3 **WREN:** Data EEPROM write enable
0: Disable
1: Enable
This is the Data EEPROM Write Enable Bit, which must be set high before Data EEPROM write operations are carried out. Clearing this bit to zero will inhibit Data EEPROM write operations. Note that the WREN bit will automatically be cleared to zero after the write operation is finished.

Bit 2 **WR:** Data EEPROM write control
0: Write cycle has finished
1: Activate a write cycle
This is the Data EEPROM Write Control Bit. When this bit is set high by the application program, a write cycle will be activated. This bit will be automatically reset to zero by hardware after the write cycle has finished. Setting this bit high will have no effect if the WREN has not first been set high.

Bit 1 **RDEN:** Data EEPROM read enable
0: Disable
1: Enable
This is the Data EEPROM Read Enable Bit, which must be set high before Data EEPROM read operations are carried out. Clearing this bit to zero will inhibit Data EEPROM read operations.

Bit 0 **RD:** Data EEPROM read control
0: Read cycle has finished
1: Activate a read cycle
This is the Data EEPROM Read Control Bit. When this bit is set high by the application program, a read cycle will be activated. This bit will be automatically reset to zero by hardware after the read cycle has finished. Setting this bit high will have no effect if the RDEN has not first been set high.

Note: 1. The EREN, ER, WREN, WR, RDEN and RD cannot be set to "1" at the same time in one instruction. The WR and RD cannot be set to "1" at the same time.
2. Ensure that the f_{SUB} clock is stable before executing the erase or write operation.
3. Ensure that the erase or write operation is totally complete before changing the contents of the EEPROM related registers or activating the IAP function.

Read Operation from the EEPROM

Reading data from the EEPROM can be implemented by two modes for the device, byte read mode or page read mode, which is controlled by the EEPROM operation mode selection bit, MODE, in the EEC register.

Byte Read Mode

The EEPROM byte read operation can be executed when the mode selection bit, MODE, is cleared to zero. For a byte read operation the desired EEPROM address should first be placed in the EEAH and EEAL registers, as well as the read enable bit, RDEN, in the EEC register should be set high to enable the read function. Then setting the RD bit high will initiate the EEPROM byte read operation. Note that setting the RD bit high only will not initiate a read operation if the RDEN bit is not set high. When the read cycle terminates, the RD bit will automatically be cleared to zero

and the EEPROM data can be read from the EED register. The data will remain in the EED register until another read or write operation is executed. The application program can poll the RD bit to determine when the data is valid for reading.

Page Read Mode

The EEPROM page read operation can be executed when the mode selection bit, MODE, is set high. The page size can be up to 16 bytes for the page read operation. For a page read operation the start address of the desired EEPROM page should first be placed in the EEAH and EEAL registers, as well as the read enable bit, RDEN, in the EEC register should be set high to enable the read function. Then setting the RD bit high will initiate the EEPROM page read operation. Note that setting the RD bit high only will not initiate a read operation if the RDEN bit is not set high. When the current byte read cycle terminates, the RD bit will automatically be cleared indicating that the EEPROM data can be read from the EED register, and the current address will be incremented by one by hardware. The data which is stored in the next EEPROM address can continuously be read when the RD bit is again set high without reconfiguring the EEPROM address and RDEN control bit. The application program can poll the RD bit to determine when the data is valid for reading.

The EEPROM address higher 6 bits are used to specify the desired page location while the lower 4 bits are used to point to the actual address. In the page read operation mode the lower 4-bit address value will automatically be incremented by one. However, the higher 6-bit address value will not be incremented by hardware. When the EEPROM address lower 4-bit value which is internally incremented by one in the page mode reaches the page boundary, known as 0FH, the EEPROM address lower 4-bit value will stop at 0FH. The EEPROM address will not “roll over”.

Page Erase Operation to the EEPROM

The EEPROM page erase operation can be executed when the mode selection bit, MODE, is set high. The EEPROM is capable of a 16-byte page erase. The internal page buffer will be cleared by hardware after power on reset. When the EEPROM erase enable control bit, namely EREN, is changed from “1” to “0”, the internal page buffer will also be cleared. Note that when the EREN bit is changed from “0” to “1”, the internal page buffer will not be cleared. The EEPROM address higher 6 bits are used to specify the desired page location while the lower 4 bits are used to point to the actual address. In the page erase operation mode the lower 4-bit address value will automatically be incremented by one after each dummy data byte is written into the EED register. However, the higher 6-bit address value will not be incremented by hardware. When the EEPROM address lower 4-bit value which is internally incremented by one in the page mode reaches the page boundary, known as 0FH, the EEPROM address lower 4-bit value will stop at 0FH. The EEPROM address will not “roll over”.

For page erase operations the start address of the desired EEPROM page should first be placed in the EEAH and EEAL registers and the dummy data to be written should be placed in the EED register. The maximum data length for a page is 16 bytes. Note that the write operation to the EED register is used to tag address, it must be implemented to determine which addresses to be erased. When the page dummy data is completely written, then the EREN bit in the EEC register should be set high to enable erase operations and the ER bit must be immediately set high to initiate the EEPROM erase process. These two instructions must be executed in two consecutive instruction cycles to activate an erase operation successfully. The global interrupt enable bit EMI should also first be cleared before implementing an erase operation and then set again after a valid erase activation procedure has completed.

As the EEPROM erase cycle is controlled using an internal timer whose operation is asynchronous to microcontroller system clock, a certain time will elapse before the data will have been erased from

the EEPROM. Detecting when the erase cycle has finished can be implemented either by polling the ER bit in the EEC register or by using the EEPROM interrupt. When the erase cycle terminates, the ER bit will be automatically cleared to zero by the microcontroller, informing the user that the page data has been erased. The application program can therefore poll the ER bit to determine when the erase cycle has ended. After the erase operation is finished, the EREN bit will be cleared to zero by hardware. The Data EEPROM erased page content will all be zero after a page erase operation.

Write Operation to the EEPROM

Writing data to the EEPROM can be implemented by two modes for the device, byte write mode or page write mode, which is controlled by the EEPROM operation mode selection bit, MODE, in the EEC register.

Byte Write Mode

The EEPROM byte write operation can be executed when the mode selection bit, MODE, is cleared to zero. For byte write operations the desired EEPROM address should first be placed in the EEAH and EEAL registers and the data to be written should be placed in the EED register. To write data to the EEPROM, the write enable bit, WREN, in the EEC register must first be set high to enable the write function. After this, the WR bit in the EEC register must be immediately set high to initiate a write cycle. These two instructions must be executed in two consecutive instruction cycles to activate a write operation successfully. The global interrupt bit EMI should also first be cleared before implementing any write operations, and then set high again after a valid write activation procedure has completed. Note that setting the WR bit high only will not initiate a write cycle if the WREN bit is not set.

As the EEPROM write cycle is controlled using an internal timer whose operation is asynchronous to microcontroller system clock, a certain time will elapse before the data will have been written into the EEPROM. Detecting when the write cycle has finished can be implemented either by polling the WR bit in the EEC register or by using the EEPROM interrupt. When the write cycle terminates, the WR bit will be automatically cleared to zero by the microcontroller, informing the user that the data has been written to the EEPROM. The application program can therefore poll the WR bit to determine when the write cycle has ended. After the write operation is finished, the WREN bit will be cleared to zero by hardware. Note that a byte erase operation will automatically be executed before a byte write operation is successfully activated.

Page Write Mode

Before a page write operation is executed, it is important to ensure that a relevant page erase operation has been successfully executed. The EEPROM page write operation can be executed when the mode selection bit, MODE, is set high. The EEPROM is capable of a 16-byte page write. The internal page buffer will be cleared by hardware after power on reset. When the EEPROM write enable control bit, namely WREN, is changed from “1” to “0”, the internal page buffer will also be cleared. Note that when the WREN bit is changed from “0” to “1”, the internal page buffer will not be cleared. A page write is initiated in the same way as a byte write initiation except that the EEPROM data can be written up to 16 bytes. The EEPROM address higher 6 bits are used to specify the desired page location while the lower 4 bits are used to point to the actual address. In the page write operation mode the lower 4-bit address value will automatically be incremented by one after each data byte is written into the EED register. However, the higher 6-bit address value will not be incremented by hardware. When the EEPROM address lower 4-bit value which is internally incremented by one in the page mode reaches the page boundary, known as 0FH, the EEPROM address lower 4-bit value will stop at 0FH. The EEPROM address will not “roll over”. At this point any data write operations to the EED register will be invalid.

For page write operations the start address of the desired EEPROM page should first be placed in the EEAH and EEAL registers and the data to be written should be placed in the EED register. The maximum data length for a page is 16 bytes. Note that when a data byte is written into the EED register, then the data in the EED register will be loaded into the internal page buffer and the current address value will automatically be incremented by one. When the page data is completely written into the page buffer, then the WREN bit in the EEC register should be set high to enable write operations and the WR bit must be immediately set high to initiate the EEPROM write process. These two instructions must be executed in two consecutive instruction cycles to activate a write operation successfully. The global interrupt enable bit EMI should also first be cleared before implementing any write operations, and then set high again after a valid write activation procedure has completed. Note that setting the WR bit high only will not initiate a write cycle if the WREN bit is not set.

As the EEPROM write cycle is controlled using an internal timer whose operation is asynchronous to microcontroller system clock, a certain time will elapse before the data will have been written into the EEPROM. Detecting when the write cycle has finished can be implemented either by polling the WR bit in the EEC register or by using the EEPROM interrupt. When the write cycle terminates, the WR bit will be automatically cleared to zero by the microcontroller, informing the user that the data has been written to the EEPROM. The application program can therefore poll the WR bit to determine when the write cycle has ended. After the write operation is finished, the WREN bit will be cleared to zero by hardware.

Write Protection

Protection against inadvertent write operation is provided in several ways. After the device is powered-on the Write Enable bit in the control register will be cleared preventing any write operations. Also at power-on the Memory Pointer high byte register, MP1H or MP2H, will be reset to zero, which means that Data Memory Sector 0 will be selected. As the EEPROM control register is located in Sector 1, this adds a further measure of protection against spurious write operations. During normal program operation, ensuring that the Write Enable bit in the control register is cleared will safeguard against incorrect write operations.

EEPROM Interrupt

The EEPROM erase or write interrupt is generated when an EEPROM erase or write cycle has ended. The EEPROM interrupt must first be enabled by setting the DEE bit in the relevant interrupt register. When an EEPROM erase or write cycle ends, the DEF request flag will be set. If the global and EEPROM interrupts are enabled and the stack is not full, a jump to the associated EEPROM Interrupt vector will take place. When the interrupt is serviced, the EEPROM Interrupt request flag, DEF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts. More details can be obtained in the Interrupts section.

Programming Considerations

Care must be taken that data is not inadvertently written to the EEPROM. Protection can be enhanced by ensuring that the Write Enable bit is normally cleared to zero when not writing. Also the Memory Pointer high byte register, MP1H or MP2H, could be normally cleared to zero as this would inhibit access to Sector 1 where the EEPROM control register exists. Although certainly not necessary, consideration might be given in the application program to the checking of the validity of new write data by a simple read back process.

When writing data the WR bit must be set high immediately after the WREN bit has been set high, to ensure the write cycle executes correctly. The global interrupt bit EMI should also be cleared before a write or erase cycle is executed and then set again after a valid write or erase activation procedure has completed. Note that the device should not enter the IDLE or SLEEP mode until the EEPROM read, erase or write operation is totally complete. Otherwise, the EEPROM read, erase or write operation will fail.

Programming Examples

Reading a Data Byte from the EEPROM – Polling Method

```
MOV A, 040H                 ; setup memory pointer low byte MP1L
MOV MP1L, A                 ; MP1 points to EEC register
MOV A, 01H                  ; setup memory pointer high byte MP1H
MOV MP1H, A
CLR IAR1.4                  ; clear MODE bit, select byte operation mode
MOV A, EEPROM_ADRES_H       ; user defined high byte address
MOV EEAH, A
MOV A, EEPROM_ADRES_L       ; user defined low byte address
MOV EEAL, A
SET IAR1.1                  ; set RDEN bit, enable read operations
SET IAR1.0                  ; start Read Cycle - set RD bit
BACK:
SZ  IAR1.0                  ; check for read cycle end
JMP BACK
CLR IAR1                    ; disable EEPROM read function
CLR MP1H
MOV A, EED                  ; move read data to register
MOV READ_DATA, A
```

Reading a Data Page from the EEPROM – Polling Method

```
MOV A, 040H                 ; setup memory pointer low byte MP1L
MOV MP1L, A                 ; MP1 points to EEC register
MOV A, 01H                  ; setup memory pointer high byte MP1H
MOV MP1H, A
SET IAR1.4                  ; set MODE bit, select page operation mode
MOV A, EEPROM_ADRES_H       ; user defined high byte address
MOV EEAH, A
MOV A, EEPROM_ADRES_L       ; user defined low byte address
MOV EEAL, A
SET IAR1.1                  ; set RDEN bit, enable read operations
; ~~~~ The data length can be up to 16 bytes (Start) ~~~~
CALL READ
CALL READ
:
:
JMP PAGE_READ_FINISH
; ~~~~ The data length can be up to 16 bytes (End) ~~~~
READ:
SET IAR1.0                  ; start Read Cycle - set RD bit
BACK:
SZ  IAR1.0                  ; check for read cycle end
JMP BACK
MOV A, EED                  ; move read data to register
MOV READ_DATA, A
RET
:
PAGE_READ_FINISH:
CLR IAR1                    ; disable EEPROM read function
CLR MP1H
```

Erasing a Data Page to the EEPROM – Polling Method

```
MOV A, 040H                 ; setup memory pointer low byte MP1L
MOV MP1L, A                 ; MP1 points to EEC register
MOV A, 01H                  ; setup memory pointer high byte MP1H
MOV MP1H, A
SET IAR1.4                  ; set MODE bit, select page operation mode
MOV A, EEPROM_ADRES_H       ; user defined high byte address
MOV EEAH, A
```

```
MOV A, EEPROM_ADRES_L      ; user defined low byte address
MOV EEAL, A
; ~~~~ The data length can be up to 16 bytes (Start) ~~~~
CALL WRITE_BUF
CALL WRITE_BUF
:
:
JMP Erase_START
; ~~~~ The data length can be up to 16 bytes (End) ~~~~
WRITE_BUF:
MOV A, EEPROM_DATA         ; user defined data, erase mode don't care data value
MOV EED, A
RET
:
Erase_START:
CLR EMI
SET IAR1.6                 ; set EREN bit, enable erase operations
SET IAR1.5                 ; start Erase Cycle - set ER bit - executed immediately
                           ; after setting EREN bit
SET EMI
BACK:
SZ  IAR1.5                 ; check for erase cycle end
JMP BACK
CLR MP1H
```

Writing a Data Byte to the EEPROM – Polling Method

```
MOV A, 040H                ; setup memory pointer low byte MP1L
MOV MP1L, A                ; MP1 points to EEC register
MOV A, 01H                 ; setup memory pointer high byte MP1H
MOV MP1H, A
CLR IAR1.4                 ; clear MODE bit, select byte operation mode
MOV A, EEPROM_ADRES_H      ; user defined high byte address
MOV EEAH, A
MOV A, EEPROM_ADRES_L      ; user defined low byte address
MOV EEAL, A
MOV A, EEPROM_DATA         ; user defined data
MOV EED, A
CLR EMI
SET IAR1.3                 ; set WREN bit, enable write operations
SET IAR1.2                 ; start Write Cycle - set WR bit - executed immediately
                           ; after setting WREN bit
SET EMI
BACK:
SZ  IAR1.2                 ; check for write cycle end
JMP BACK
CLR MP1H
```

Writing a Data Page to the EEPROM – Polling Method

```
MOV A, 040H                ; setup memory pointer low byte MP1L
MOV MP1L, A                ; MP1 points to EEC register
MOV A, 01H                 ; setup memory pointer high byte MP1H
MOV MP1H, A
SET IAR1.4                 ; set MODE bit, select page operation mode
MOV A, EEPROM_ADRES_H      ; user defined high byte address
MOV EEAH, A
MOV A, EEPROM_ADRES_L      ; user defined low byte address
MOV EEAL, A
; ~~~~ The data length can be up to 16 bytes (Start) ~~~~
CALL WRITE_BUF
```

```
CALL WRITE_BUF
:
:
JMP WRITE_START
; ~~~~ The data length can be up to 16 bytes (End) ~~~~
WRITE_BUF:
MOV A, EEPROM_DATA          ; user defined data
MOV EED, A
RET
:
WRITE_START:
CLR EMI
SET IAR1.3                  ; set WREN bit, enable write operations
SET IAR1.2                  ; start Write Cycle - set WR bit - executed immediately
                            ; after setting WREN bit
SET EMI
BACK:
SZ  IAR1.2                  ; check for write cycle end
JMP BACK
CLR MP1H
```

Oscillators

Various oscillator types offer the user a wide range of functions according to their various application requirements. The flexible features of the oscillator functions ensure that the best optimisation can be achieved in terms of speed and power saving. Oscillator selections and operation are selected through a combination of configuration options and relevant control registers.

Oscillator Overview

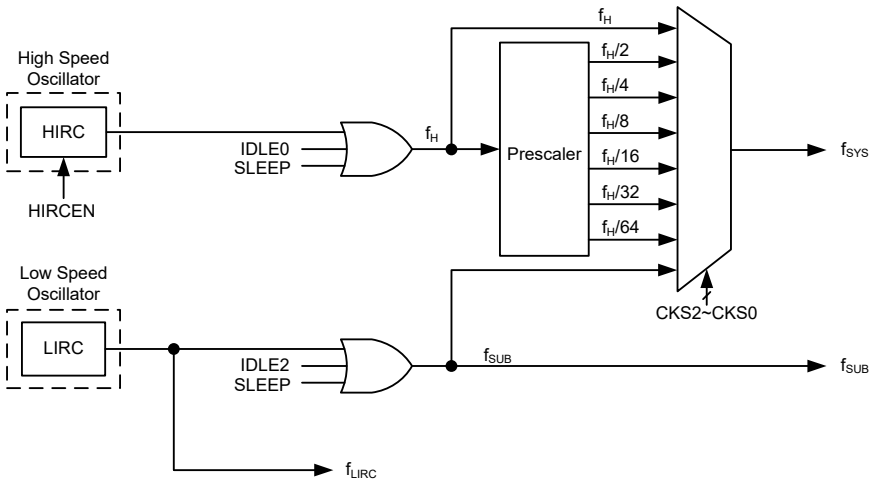
In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer and Time Base Interrupts. Fully integrated internal oscillators, requiring no external components, are provided to form a wide range of both fast and slow system oscillators. The higher frequency oscillator provides higher performance but carry with it the disadvantage of higher power requirements, while the opposite is of course true for the lower frequency oscillator. With the capability of dynamically switching between fast and slow system clock, the device has the flexibility to optimize the performance/power ratio, a feature especially important in power sensitive portable applications.

Type	Name	Frequency
Internal High Speed RC	HIRC	12/16/20MHz
Internal Low Speed RC	LIRC	32kHz

Oscillator Types

System Clock Configurations

There are two methods of generating the system clock, a high speed oscillator and a low speed oscillator for the device. The high speed oscillator is the internal 12/16/20MHz RC oscillator, HIRC. The low speed oscillator is the internal 32kHz RC oscillator, LIRC. Selecting whether the low or high speed oscillator is used as the system oscillator is implemented using the CKS2~CKS0 bits in the SCC register and as the system clock can be dynamically selected.



System Clock Configurations

Internal High Speed RC Oscillator – HIRC

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has three fixed frequencies of 12MHz, 16MHz and 20MHz, which are selected by the HIRC1~HIRC0 bits in the HIRCC register. These bits must also be setup to match the selected configuration option frequency to ensure that the HIRC frequency accuracy specified in the A.C. Characteristics is achieved. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised.

Internal 32kHz Oscillator – LIRC

The Internal 32kHz System Oscillator is a low frequency oscillator. It is also a fully integrated RC oscillator with a typical frequency of 32kHz, requiring no external components for its implementation. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised.

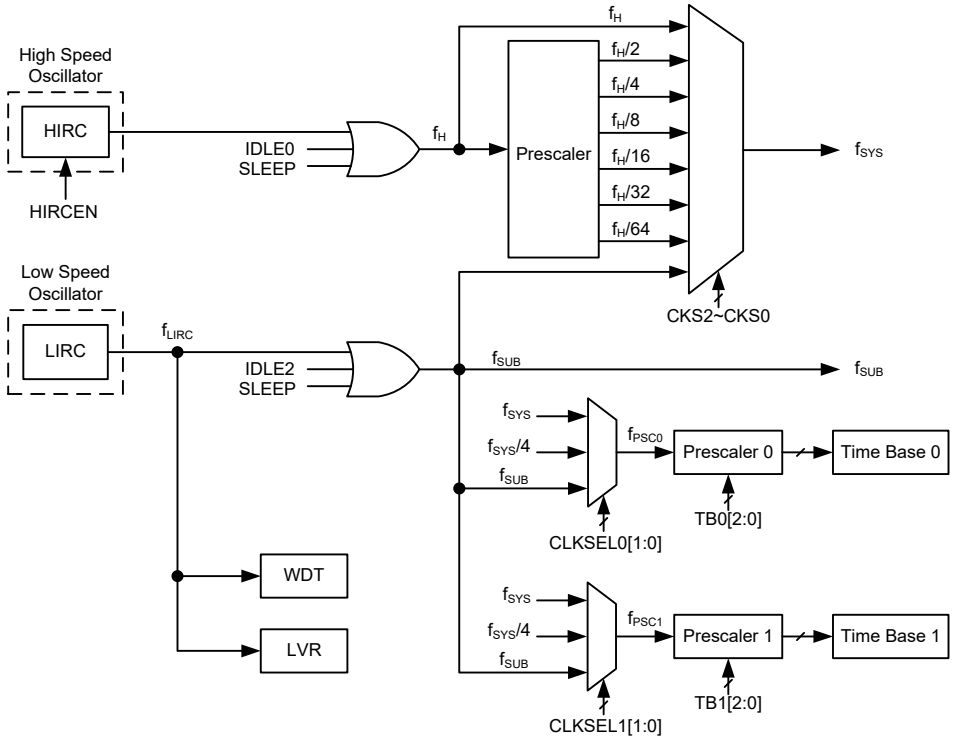
Operating Modes and System Clocks

Present day applications require that their microcontrollers have high performance but often still demand that they consume as little power as possible, conflicting requirements that are especially true in battery powered portable applications. The fast clocks required for high performance will by their nature increase current consumption and of course vice-versa lower speed clocks reduce current consumption. As Holtek has provided the device with both high and low speed clock sources and the means to switch between them dynamically, the user can optimise the operation of their microcontrollers to achieve the best performance/power ratio.

System Clocks

The device has different clock sources for both the CPU and peripheral function operation. By providing the user with a wide range of clock selections using register programming, a clock system can be configured to obtain maximum application performance.

The main system clock, can come from either a high frequency, f_H, or low frequency, f_{SUB}, source, and is selected using the CKS2~CKS0 bits in the SCC register. The high speed system clock is sourced from the HIRC oscillator. The low speed system clock source is sourced from the LIRC oscillator. The other choice, which is a divided version of the high speed system oscillator has a range of $f_H/2$~$f_H/64$.



Device Clock Configurations

Note: When the system clock source f_{SYS} is switched to f_{SUB} from f_H, the high speed oscillator can be stopped to conserve the power or continue to oscillate to provide the clock source, f_H~$f_H/64$, for peripheral circuit to use, which is determined by configuring the corresponding high speed oscillator enable control bit.

System Operation Modes

There are six different modes of operation for the microcontroller, each one with its own special characteristics and which can be chosen according to the specific performance and power requirements of the application. There are two modes allowing normal operation of the microcontroller, the FAST Mode and SLOW Mode. The remaining four modes, the SLEEP, IDLE0, IDLE1 and IDLE2 Mode are used when the microcontroller CPU is switched off to conserve power.

Operation Mode	CPU	Register Setting			f_{SYS}	f_H	f_{SUB}	f_{LIRC}
		FHIDEN	FSIDEN	CKS2~CKS0				
FAST	On	x	x	000~110	f_H~$f_H/64$	On	On	On
SLOW	On	x	x	111	f_{SUB}	On/Off (1)	On	On
IDLE0	Off	0	1	000~110	Off	Off	On	On
				111	On			
IDLE1	Off	1	1	xxx	On	On	On	On
IDLE2	Off	1	0	000~110	On	On	Off	On
				111	Off			

Operation Mode	CPU	Register Setting			f_{SYS}	f_H	f_{SUB}	f_{LIRC}
		FHIDEN	FSIDEN	CKS2~CKS0				
SLEEP	Off	0	0	xxx	Off	Off	Off	On/Off (2)

"x": don't care

Note: 1. The f_H clock will be switched on or off by configuring the corresponding oscillator enable bit in the SLOW mode.
2. The f_{LIRC} clock can be on or off which is controlled by the WDT function being enabled or disabled in the SLEEP mode.

FAST Mode

This is one of the main operating modes where the microcontroller has all of its functions operational and where the system clock is provided by the high speed oscillator. This mode operates allowing the microcontroller to operate normally with a clock source will come from the HIRC oscillator. The high speed oscillator will however first be divided by a ratio ranging from 1 to 64, the actual ratio being selected by the CKS2~CKS0 bits in the SCC register. Although a high speed oscillator is used, running the microcontroller at a divided clock ratio reduces the operating current.

SLOW Mode

This is also a mode where the microcontroller operates normally although now with a slower speed clock source. The clock source used will be from f_{SUB}. The f_{SUB} clock is derived from the LIRC oscillator.

SLEEP Mode

The SLEEP Mode is entered when a HALT instruction is executed and when the FHIDEN and FSIDEN bits are low. In the SLEEP mode the CPU will be stopped and both the high and low speed oscillators will be switched off. However the f_{LIRC} clock will continue to operate if the WDT function is enabled by the WDTC register.

IDLE0 Mode

The IDLE0 Mode is entered when a HALT instruction is executed and when the FHIDEN bit in the SCC register is low and the FSIDEN bit in the SCC register is high. In the IDLE0 Mode the CPU will be switched off but the low speed oscillator will be turned on to drive some peripheral functions.

IDLE1 Mode

The IDLE1 Mode is entered when a HALT instruction is executed and when the FHIDEN bit in the SCC register is high and the FSIDEN bit in the SCC register is high. In the IDLE1 Mode the CPU will be switched off but both the high and low speed oscillators will be turned on to provide a clock source to keep some peripheral functions operational.

IDLE2 Mode

The IDLE2 Mode is entered when a HALT instruction is executed and when the FHIDEN bit in the SCC register is high and the FSIDEN bit in the SCC register is low. In the IDLE2 Mode the CPU will be switched off but the high speed oscillator will be turned on to provide a clock source to keep some peripheral functions operational.

Control Registers

The SCC and HIRCC registers are used to control the system clock and the corresponding oscillator configurations.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SCC	CKS2	CKS1	CKS0	—	—	—	FHIDEN	FSIDEN
HIRCC	—	—	—	—	HIRC1	HIRC0	HIRCF	HIRCEN

System Operating Mode Control Register List

• SCC Register

Bit	7	6	5	4	3	2	1	0
Name	CKS2	CKS1	CKS0	—	—	—	FHIDEN	FSIDEN
R/W	R/W	R/W	R/W	—	—	—	R/W	R/W
POR	0	1	0	—	—	—	0	0

Bit 7~5 **CKS2~CKS0:** System clock selection
000: f_H
001: $f_H/2$
010: $f_H/4$
011: $f_H/8$
100: $f_H/16$
101: $f_H/32$
110: $f_H/64$
111: f_{SUB}

These three bits are used to select which clock is used as the system clock source. In addition to the system clock source directly derived from f_H or f_{SUB}, a divided version of the high speed system oscillator can also be chosen as the system clock source.

Bit 4~2 Unimplemented, read as "0"

Bit 1 **FHIDEN:** High Frequency oscillator control when CPU is switched off
0: Disable
1: Enable

This bit is used to control whether the high speed oscillator is activated or stopped when the CPU is switched off by executing a "HALT" instruction.

Bit 0 **FSIDEN:** Low Frequency oscillator control when CPU is switched off
0: Disable
1: Enable

This bit is used to control whether the low speed oscillator is activated or stopped when the CPU is switched off by executing a "HALT" instruction.

Note: A certain delay is required before the relevant clock is successfully switched to the target clock source after any clock switching setup using the CKS2~CKS0 bits. A proper delay time must be arranged before executing the following operations which require immediate reaction with the target clock source.

Clock switching delay time = $4\times t_{SYS}+[0\sim(1.5\times t_{Curr.}+0.5\times t_{Tar.})]$, where $t_{Curr.}$ indicates the current clock period, $t_{Tar.}$ indicates the target clock period and t_{SYS} indicates the current system clock period.

• **HIRCC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	HIRC1	HIRC0	HIRCF	HIRCEN
R/W	—	—	—	—	R/W	R/W	R	R/W
POR	—	—	—	—	0	0	0	1

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **HIRC1~HIRC0:** HIRC frequency selection
00: 12MHz
01: 16MHz
10: 20MHz
11: 12MHz

When the HIRC oscillator is enabled or the HIRC frequency selection is changed by the application program, the clock frequency will automatically be changed after the HIRCF flag is set to 1.

It is recommended that the HIRC frequency selected by these two bits should be the same with the frequency determined by the configuration option to achieve the HIRC frequency accuracy specified in the A.C. Characteristics.

Bit 1 **HIRCF:** HIRC oscillator stable flag
0: HIRC unstable
1: HIRC stable

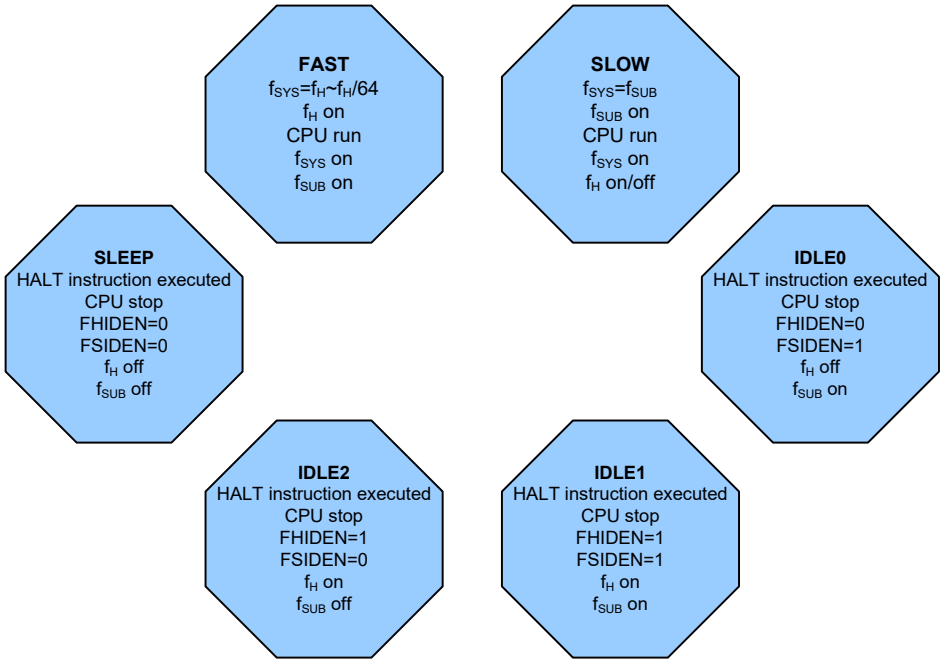
This bit is used to indicate whether the HIRC oscillator is stable or not. When the HIRCEN bit is set to 1 to enable the HIRC oscillator or the HIRC frequency selection is changed by the application program, the HIRCF bit will first be cleared to 0 and then set to 1 after the HIRC oscillator is stable.

Bit 0 **HIRCEN:** HIRC oscillator enable control
0: Disable
1: Enable

Operating Mode Switching

The device can switch between operating modes dynamically allowing the user to select the best performance/power ratio for the present task in hand. In this way microcontroller operations that do not require high performance can be executed using slower clocks thus requiring less operating current and prolonging battery life in portable applications.

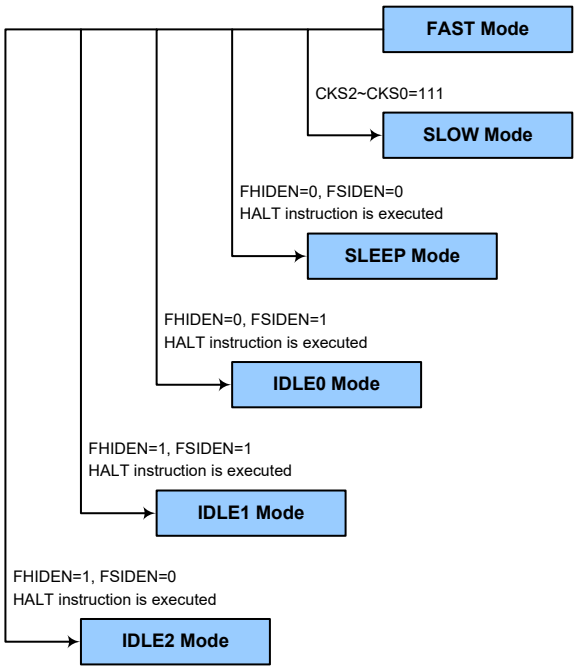
In simple terms, Mode Switching between the FAST Mode and SLOW Mode is executed using the CKS2~CKS0 bits in the SCC register while Mode Switching from the FAST/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When a HALT instruction is executed, whether the device enters the IDLE Mode or the SLEEP Mode is determined by the condition of the FHIDEN and FSIDEN bits in the SCC register.



FAST Mode to SLOW Mode Switching

When running in the FAST Mode, which uses the high speed system oscillator, and therefore consumes more power, the system clock can switch to run in the SLOW Mode by setting the CKS2~CKS0 bits to "111" in the SCC register. This will then use the low speed system oscillator which will consume less power. Users may decide to do this for certain operations which do not require high performance and can subsequently reduce power consumption.

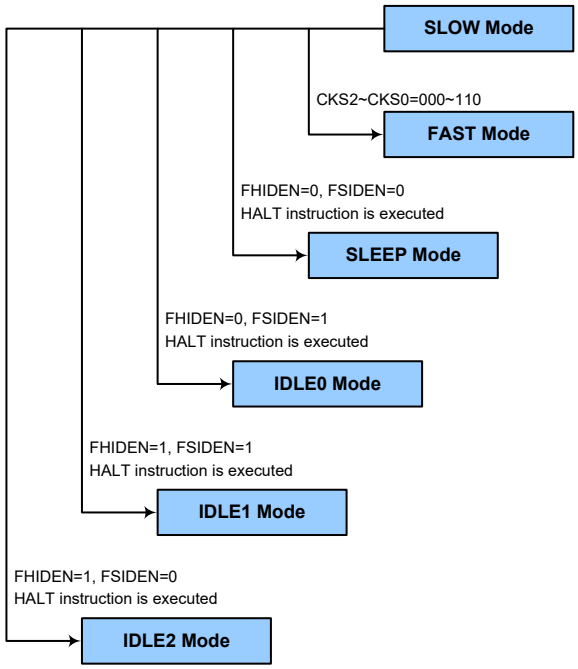
The SLOW Mode is sourced from the LIRC oscillator and therefore requires the oscillator to be stable before full mode switching occurs.



SLOW Mode to FAST Mode Switching

In SLOW mode the system clock is derived from f_{SUB}. When system clock is switched back to the FAST mode from f_{SUB}, the CKS2~CKS0 bits should be set to "000"~"110" and then the system clock will respectively be switched to f_H~f_H/64.

However, if f_H is not used in SLOW mode and thus switched off, it will take some time to re-oscillate and stabilise when switching to the FAST mode from the SLOW Mode. This is monitored using the HIRCF bit in the HIRCC register. The time duration required for the high speed system oscillator stabilization is specified in the System Start Up Time Characteristics.



Entering the SLEEP Mode

There is only one way for the device to enter the SLEEP Mode and that is to execute the "HALT" instruction in the application program with both the FHIDEN and FSIDEN bits in the SCC register equal to "0". In this mode all the clocks and functions will be switched off except the WDT function. When this instruction is executed under the conditions described above, the following will occur:

- The system clock will be stopped and the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag PDF will be set, and WDT timeout flag TO will be cleared.
- The WDT will be cleared and resume counting if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

Entering the IDLE0 Mode

There is only one way for the device to enter the IDLE0 Mode and that is to execute the "HALT" instruction in the application program with the FHIDEN bit in the SCC register equal to "0" and the FSIDEN bit in the SCC register equal to "1". When this instruction is executed under the conditions described above, the following will occur:

- The f_H clock will be stopped and the application program will stop at the "HALT" instruction, but the f_{SUB} clock will be on.

- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag PDF will be set, and WDT timeout flag TO will be cleared.
- The WDT will be cleared and resume counting if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

Entering the IDLE1 Mode

There is only one way for the device to enter the IDLE1 Mode and that is to execute the "HALT" instruction in the application program with both the FHIDEN and FSIDEN bits in the SCC register equal to "1". When this instruction is executed under the conditions described above, the following will occur:

- The f_H and f_{SUB} clocks will be on but the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag PDF will be set, and WDT timeout flag TO will be cleared.
- The WDT will be cleared and resume counting if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

Entering the IDLE2 Mode

There is only one way for the device to enter the IDLE2 Mode and that is to execute the "HALT" instruction in the application program with the FHIDEN bit in the SCC register equal to "1" and the FSIDEN bit in the SCC register equal to "0". When this instruction is executed under the conditions described above, the following will occur:

- The f_H clock will be on but the f_{SUB} clock will be off and the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag PDF will be set, and WDT timeout flag TO will be cleared.
- The WDT will be cleared and resume counting if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

Standby Current Considerations

As the main reason for entering the SLEEP or IDLE Mode is to keep the current consumption of the device to as low a value as possible, perhaps only in the order of several micro-amps except in the IDLE1 and IDLE2 Mode, there are other considerations which must also be taken into account by the circuit designer if the power consumption is to be minimised. Special attention must be made to the I/O pins on the device. All high-impedance input pins must be connected to either a fixed high or low level as any floating input pins could create internal oscillations and result in increased current consumption. This also applies to the device which has different package types, as there may be unbonded pins. These must either be setup as outputs or if setup as inputs must have pull-high resistors connected.

Care must also be taken with the loads, which are connected to I/O pins, which are setup as outputs. These should be placed in a condition in which minimum current is drawn or connected only to external circuits that do not draw current, such as other CMOS inputs. Also note that additional standby current will also be required if the LIRC oscillator has enabled.

In the IDLE1 and IDLE2 Mode the high speed oscillator is on, if the peripheral function clock source is derived from the high speed oscillator, the additional standby current will also be perhaps in the order of several hundred micro-amps.

Wake-up

To minimise power consumption the device can enter the SLEEP or any IDLE Mode, where the CPU will be switched off. However, when the device is woken up again, it will take a considerable time for the original system oscillator to restart, stabilise and allow normal operation to resume.

After the system enters the SLEEP or IDLE Mode, it can be woken up from one of various sources listed as follows:

- An external falling edge on Port A
- A system interrupt
- A WDT overflow

When the device executes the "HALT" instruction, it will enter the IDLE or SLEEP mode and the PDF flag will be set to 1. The PDF flag will be cleared to 0 if the device experiences a system power-up or executes the clear Watchdog Timer instruction. If the system is woken up by a WDT overflow, a Watchdog Timer reset will be initiated and the TO flag will be set to 1. The TO flag is set if a WDT time-out occurs and causes a wake-up that only resets the Program Counter and Stack Pointer, other flags remain in their original status.

Each pin on Port A can be setup using the PAWU register to permit a negative transition on the pin to wake up the system. When a Port A pin wake-up occurs, the program will resume execution at the instruction following the "HALT" instruction. If the system is woken up by an interrupt, then two possible situations may occur. The first is where the related interrupt is disabled or the interrupt is enabled but the stack is full, in which case the program will resume execution at the instruction following the "HALT" instruction. In this situation, the interrupt which woke up the device will not be immediately serviced, but will rather be serviced later when the related interrupt is finally enabled or when a stack level becomes free. The other situation is where the related interrupt is enabled and the stack is not full, in which case the regular interrupt response takes place. If an interrupt request flag is set high before entering the SLEEP or IDLE Mode, the wake-up function of the related interrupt will be disabled.

Watchdog Timer

The Watchdog Timer is provided to prevent program malfunctions or sequences from jumping to unknown locations, due to certain uncontrollable external events such as electrical noise.

Watchdog Timer Clock Source

The Watchdog Timer clock source is provided by the internal RC oscillator, f_{LIRC}. The LIRC internal oscillator has an approximate frequency of 32kHz and this specified internal clock period can vary with V_{DD}, temperature and process variations. The Watchdog Timer source clock is then subdivided by a ratio of 2^8 to 2^{18} to give longer timeouts, the actual value being chosen using the WS2~WS0 bits in the WDTC register.

Watchdog Timer Control Register

A single register, WDTC, controls the required timeout period as well as the WDT enable/disable and software reset MCU operation. This register controls the overall operation of the Watchdog Timer.

• **WDTC Register**

Bit	7	6	5	4	3	2	1	0
Name	WE4	WE3	WE2	WE1	WE0	WS2	WS1	WS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	1	0	1	0	0	1	1

Bit 7~3 **WE4~WE0:** WDT function enable control
10101: Disable
01010: Enable
Other values: Reset MCU
If these bits are changed due to adverse environmental conditions, the microcontroller will be reset. The reset operation will be activated after a delay time, t_{SRESET}, and the WRF bit in the RSTFC register will be set to 1.

Bit 2~0 **WS2~WS0:** WDT time-out period selection
000: $2^8/f_{LIRC}$
001: $2^{10}/f_{LIRC}$
010: $2^{12}/f_{LIRC}$
011: $2^{14}/f_{LIRC}$
100: $2^{15}/f_{LIRC}$
101: $2^{16}/f_{LIRC}$
110: $2^{17}/f_{LIRC}$
111: $2^{18}/f_{LIRC}$
These three bits determine the division ratio of the watchdog timer source clock, which in turn determines the time-out period.

• **RSTFC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	LVRF	LRF	WRF
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	x	0	0

"x": unknown

Bit 7~3 Unimplemented, read as "0"

Bit 2 **LVRF:** LVR function reset flag
Described elsewhere.

Bit 1 **LRF:** LVR control register software reset flag
Described elsewhere.

Bit 0 **WRF:** WDT control register software reset flag
0: Not occurred
1: Occurred
This bit is set to 1 by the WDT control register software reset and cleared by the application program. Note that this bit can only be cleared to 0 by the application program.

Watchdog Timer Operation

The Watchdog Timer operates by providing a device reset when its timer overflows. This means that in the application program and during normal operation the user has to strategically clear the Watchdog Timer before it overflows to prevent the Watchdog Timer from executing a reset. This is done using the clear watchdog instruction. If the program malfunctions for whatever reason, jumps to an unknown location, or enters an endless loop, the clear instruction will not be executed in the correct manner, in which case the Watchdog Timer will overflow and reset the device. With regard to the Watchdog Timer enable/disable function, there are five bits, WE4~WE0, in the WDTC register to offer the enable/disable control and reset control of the Watchdog Timer. The WDT function will be enabled when the WE4~WE0 bits are set to a value of 01010B while the WDT function will

be disabled if the WE4~WE0 bits are equal to 10101B. If the WE4~WE0 bits are set to any other values rather than 01010B and 10101B, it will reset the device after a delay time, t_{SRESET}. After power on these bits will have a value of 01010B.

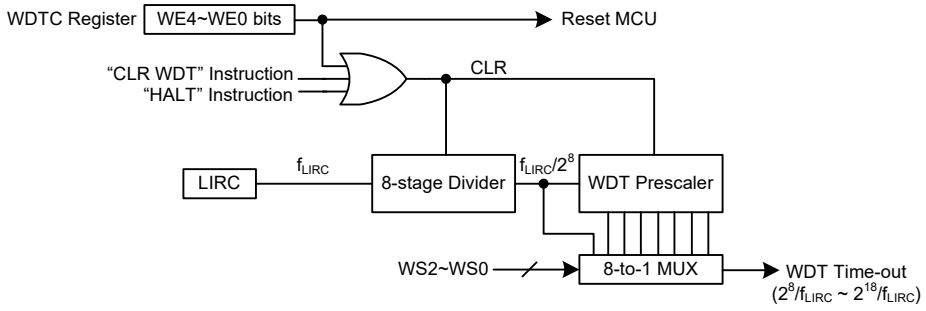
WE4~WE0 Bits	WDT Function
10101B	Disable
01010B	Enable
Any other value	Reset MCU

Watchdog Timer Function Control

Under normal program operation, a Watchdog Timer time-out will initialise a device reset and set the status bit TO. However, if the system is in the SLEEP or IDLE Mode, when a Watchdog Timer time-out occurs, the TO bit in the status register will be set and only the Program Counter and Stack Pointer will be reset. Three methods can be adopted to clear the contents of the Watchdog Timer. The first is a WDT software reset, which means a certain value except 01010B and 10101B written into the WE4~WE0 field, the second is using the Watchdog Timer software clear instruction and the third is via a HALT instruction.

There is only one method of using software instruction to clear the Watchdog Timer. That is to use the single "CLR WDT" instruction to clear the WDT contents.

The maximum time out period is when the 2^{18} division ratio is selected. As an example, with a 32kHz LIRC oscillator as its source clock, this will give a maximum watchdog period of around 8 second for the 2^{18} division ratio and a minimum timeout of 8ms for the 2^8 division ration.



Watchdog Timer

Reset and Initialisation

A reset function is a fundamental part of any microcontroller ensuring that the device can be set to some predetermined condition irrespective of outside parameters. The most important reset condition is after power is first applied to the microcontroller. In this case, internal circuitry will ensure that the microcontroller, after a short delay, will be in a well-defined state and ready to execute the first program instruction. After this power-on reset, certain important internal registers will be set to defined states before the program commences. One of these registers is the Program Counter, which will be reset to zero forcing the microcontroller to begin program execution from the lowest Program Memory address.

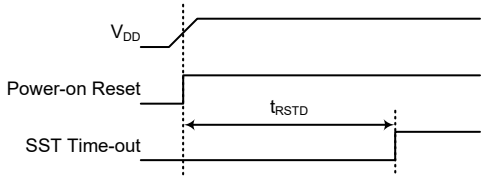
Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset is implemented in situations where the power supply voltage falls below a certain threshold. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All types of reset operations result in different register conditions being setup.

Reset Functions

There are several ways in which a microcontroller reset can occur through events occurring internally.

Power-on Reset

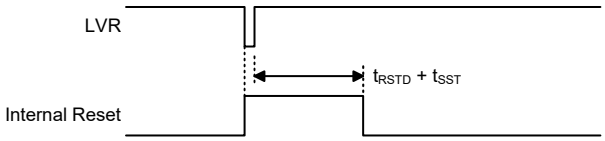
The most fundamental and unavoidable reset is the one that occurs after power is first applied to the microcontroller. As well as ensuring that the Program Memory begins execution from the first memory address, a power-on reset also ensures that certain other registers are preset to known conditions. All the I/O port and port control registers will power up in a high condition ensuring that all pins will be first set to inputs.



Power-On Reset Timing Chart

Low Voltage Reset – LVR

The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device and provides an MCU reset should the value fall below a certain predefined level. The LVR function can be enabled or disabled by the LVRC control register. If the LVRC control register is configured to select a certain defined LVR voltage, the LVR function will be always enabled in the FAST or SLOW Mode. If the supply voltage of the device drops to within a range of 0.9V~V_{LVR} such as might occur when changing the battery, the LVR will automatically reset the device internally and the LVRF bit in the RSTFC register will also be set to 1. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between 0.9V~V_{LVR} must exist for a time greater than that specified by t_{LVR} in the LVD/LVR Electrical Characteristics. If the duration of the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. The actual V_{LVR} value can be selected by the LVS7~LVS0 bits in the LVRC register. If the LVS7~LVS0 bits have any other value, which may perhaps occur due to adverse environmental conditions such as noise, the LVR will reset the device after a delay time, t_{SRESET}. When this happens, the LRF bit in the RSTFC register will be set to 1. After power on the register will have the value of 01010101B. Note that the LVR function will be automatically disabled when the device enters the SLEEP or IDLE mode.



Low Voltage Reset Timing Chart

- **LVRC Register**

Bit	7	6	5	4	3	2	1	0
Name	LVS7	LVS6	LVS5	LVS4	LVS3	LVS2	LVS1	LVS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	1	0	1	0	1	0	1

Bit 7~0 **LVS7~LVS0:** LVR voltage select
01010101: 2.1V
00110011: 2.55V
10011001: 3.15V

10101010: 3.8V
11110000: LVR disable
Other values: Generates a MCU reset – register is reset to POR value

When an actual low voltage condition occurs, as specified by one of the four defined LVR voltage value above, an MCU reset will generated. The reset operation will be activated after the low voltage condition keeps more than a t_{LVR} time. In this situation the register contents will remain the same after such a reset occurs.

Any register value, other than 11110000B and the four defined register values above, will also result in the generation of an MCU reset. The reset operation will be activated after a delay time, t_{SRESET}. However in this situation the register contents will be reset to the POR value.

- **RSTFC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	LVRF	LRF	WRF
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	x	0	0

"x": unknown

Bit 7~3 Unimplemented, read as "0"

Bit 2 **LVRF:** LVR function reset flag
0: Not occurred
1: Occurred
This bit is set to 1 when a specific low voltage reset condition occurs. Note that this bit can only be cleared to 0 by the application program.

Bit 1 **LRF:** LVR control register software reset flag
0: Not occurred
1: Occurred
This bit is set to 1 by the LVRC control register contains any undefined LVR voltage register values. This in effect acts like a software-reset function. Note that this bit can only be cleared to 0 by the application program.

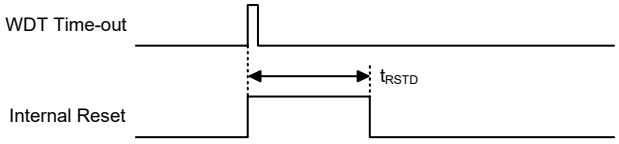
Bit 0 **WRF:** WDT control register software reset flag
Described elsewhere.

IAP Reset

When a specific value of "55H" is written into the FC1 register, a reset signal will be generated to reset the whole device. Refer to the IAP section for more associated details.

Watchdog Time-out Reset during Normal Operation

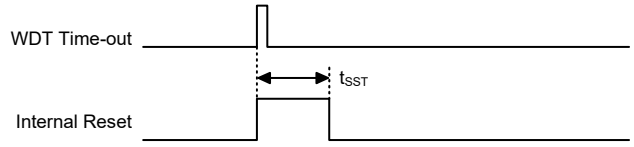
The Watchdog time-out Reset during normal operation in the FAST or SLOW Mode is the same as the hardware Low Voltage Reset except that the Watchdog time-out flag TO will be set to "1".



WDT Time-out Reset during Normal Operation Timing Chart

Watchdog Time-out Reset during SLEEP or IDLE Mode

The Watchdog time-out Reset during SLEEP or IDLE Mode is a little different from other kinds of reset. Most of the conditions remain unchanged except that the Program Counter and the Stack Pointer will be cleared to "0" and the TO and PDF flags will be set to "1". Refer to the System Start Up Time Characteristics for t_{SST} details.



WDT Time-out Reset during SLEEP or IDLE Mode Timing Chart

Reset Initial Conditions

The different types of reset described affect the reset flags in different ways. These flags, known as PDF and TO are located in the status register and are controlled by various microcontroller operations, such as the SLEEP or IDLE Mode function or Watchdog Timer. The reset flags are shown in the table:

TO	PDF	RESET Conditions
0	0	Power-on reset
u	u	LVR reset during FAST or SLOW Mode operation
1	u	WDT time-out reset during FAST or SLOW Mode operation
1	1	WDT time-out reset during IDLE or SLEEP Mode operation

"u": unchanged

The following table indicates the way in which the various components of the microcontroller are affected after a power-on reset occurs.

Item	Condition After RESET
Program Counter	Reset to zero
Interrupts	All interrupts will be disabled
WDT, Time Bases	Clear after reset, WDT begins counting
Timer Modules	Timer Modules will be turned off
Input/Output Ports	I/O ports will be setup as inputs
Stack Pointer	Stack Pointer will point to the top of the stack

The different kinds of resets all affect the internal registers of the microcontroller in different ways. To ensure reliable continuation of normal program execution after a reset occurs, it is important to know what condition the microcontroller is in after a particular reset occurs. The following table describes how each type of reset affects the microcontroller internal registers.

Register	Power On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
IAR0	0000 0000	0000 0000	uuuu uuuu
MP0	0000 0000	0000 0000	uuuu uuuu
IAR1	0000 0000	0000 0000	uuuu uuuu
MP1L	0000 0000	0000 0000	uuuu uuuu
MP1H	0000 0000	0000 0000	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBHP	--xx xxxx	--uu uuuu	--uu uuuu
STATUS	xx00 xxxx	uu1u uuuu	uu11 uuuu
PBP	---- ---0	---- ---0	---- ---u
IAR2	0000 0000	0000 0000	uuuu uuuu
MP2L	0000 0000	0000 0000	uuuu uuuu
MP2H	0000 0000	0000 0000	uuuu uuuu
RSTFC	---- -x00	---- -uuu	---- -uuu

Register	Power On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
LVRC	0101 0101	0101 0101	uuuu uuuu
LVDC	--00 -000	--00 -000	--uu -uuu
SCC	010- --00	010- --00	uuu- --uu
HIRCC	---- 0001	---- 0001	---- uuuu
PA	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	uuuu uuuu
PAPU	0000 0000	0000 0000	uuuu uuuu
PAWU	0000 0000	0000 0000	uuuu uuuu
PB	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	uuuu uuuu
PBPU	0000 0000	0000 0000	uuuu uuuu
PC	1111 1111	1111 1111	uuuu uuuu
PCC	1111 1111	1111 1111	uuuu uuuu
PCPU	0000 0000	0000 0000	uuuu uuuu
PDC	---- --11	---- --11	---- --uu
PDPUS0	---- 0000	---- 0000	---- uuuu
PAS0	00-- 00--	00-- 00--	uu-- uu--
PAS1	0000 0000	0000 0000	uuuu uuuu
PBS0	---- --00	---- --00	---- --uu
PBS1	0000 0000	0000 0000	uuuu uuuu
PCS0	0000 0000	0000 0000	uuuu uuuu
PCS1	0000 0000	0000 0000	uuuu uuuu
PDS0	---- 0000	---- 0000	---- uuuu
IECC	0000 0000	0000 0000	uuuu uuuu
WDTC	0101 0011	0101 0011	uuuu uuuu
TB0C	0--- -000	0--- -000	u--- -uuu
TB1C	0--- -000	0--- -000	u--- -uuu
SADOL	xxxx ----	xxxx ----	uuuu ---- (ADRFS=0)
			uuuu uuuu (ADRFS=1)
SADOH	xxxx xxxx	xxxx xxxx	uuuu uuuu (ADRFS=0)
			---- uuuu (ADRFS=1)
SADC0	0000 0000	0000 0000	uuuu uuuu
SADC1	0000 -000	0000 -000	uuuu -uuu
SADC2	0--0 0000	0--0 0000	u--u uuuu
VBGRC	---- ---0	---- ---0	---- ---u
INTC0	-000 0000	-000 0000	-uuu uuuu
INTC1	0000 0000	0000 0000	uuuu uuuu
INTC2	0000 0000	0000 0000	uuuu uuuu
INTC3	0000 0000	0000 0000	uuuu uuuu
MFI0	0000 0000	0000 0000	uuuu uuuu
MFI1	0000 0000	0000 0000	uuuu uuuu
MFI2	--00 --00	--00 --00	--uu --uu
INTEG	0000 0000	0000 0000	uuuu uuuu
PSC0R	---- --00	---- --00	---- --uu
PSC1R	---- --00	---- --00	---- --uu
PMPS	---- 0000	---- 0000	---- uuuu
SLEDC0	0000 0000	0000 0000	uuuu uuuu

Register	Power On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
SLEDC1	---- 0000	---- 0000	---- uuuu
ADUDA0	0000 0000	0000 0000	uuuu uuuu
ADUDA1	0000 0000	0000 0000	uuuu uuuu
ADUC0	--00 --00	--00 --00	--uu --uu
ADUC1	--00 0000	--00 0000	--uu uuuu
FC0	0000 0000	0000 0000	uuuu uuuu
FC1	0000 0000	0000 0000	uuuu uuuu
FC2	---- --00	---- --00	---- --uu
FARL	0000 0000	0000 0000	uuuu uuuu
FARH	--00 0000	--00 0000	--uu uuuu
FD0L	0000 0000	0000 0000	uuuu uuuu
FD0H	0000 0000	0000 0000	uuuu uuuu
FD1L	0000 0000	0000 0000	uuuu uuuu
FD1H	0000 0000	0000 0000	uuuu uuuu
FD2L	0000 0000	0000 0000	uuuu uuuu
FD2H	0000 0000	0000 0000	uuuu uuuu
FD3L	0000 0000	0000 0000	uuuu uuuu
FD3H	0000 0000	0000 0000	uuuu uuuu
IICMC0	---- 000-	---- 000-	---- uuu-
IICMC1	xx00 0010	xx00 0010	uuuu uuuu
IICMC2	0000 0000	0000 0000	uuuu uuuu
IICMD	xxxx xxxx	xxxx xxxx	uuuu uuuu
IICMA	0000 000-	0000 000-	uuuu uuu-
IICMC	0000 0000	0000 0000	uuuu uuuu
EEAL	0000 0000	0000 0000	uuuu uuuu
EEAH	---- --00	---- --00	---- --uu
EED	0000 0000	0000 0000	uuuu uuuu
MDUWR0	xxxx xxxx	0000 0000	uuuu uuuu
MDUWR1	xxxx xxxx	0000 0000	uuuu uuuu
MDUWR2	xxxx xxxx	0000 0000	uuuu uuuu
MDUWR3	xxxx xxxx	0000 0000	uuuu uuuu
MDUWR4	xxxx xxxx	0000 0000	uuuu uuuu
MDUWR5	xxxx xxxx	0000 0000	uuuu uuuu
MDUWCTRL	00-- ----	00-- ----	uu-- ----
OPC0	---- 0000	---- 0000	---- uuuu
OPC1	0--0 0000	0--0 0000	u--u uuuu
OPOCAL	0010 0000	0010 0000	uuuu uuuu
CRCCR	---- ---0	---- ---0	---- ---u
CRCIN	0000 0000	0000 0000	uuuu uuuu
CRCDL	0000 0000	0000 0000	uuuu uuuu
CRCDH	0000 0000	0000 0000	uuuu uuuu
SIMC0	111- 0000	111- 0000	uuu- --uu
SIMC1	1000 0001	1000 0001	uuuu uuuu
SIMD	xxxx xxxx	xxxx xxxx	uuuu uuuu
SIMA/SIMC2	0000 0000	0000 0000	uuuu uuuu
SIMTOC	0000 0000	0000 0000	uuuu uuuu
USR	0000 1011	0000 1011	uuuu uuuu
UCR1	0000 00x0	0000 00x0	uuuu uuuu
UCR2	0000 0000	0000 0000	uuuu uuuu
UCR3	---- ---0	---- ---0	---- ---u

Register	Power On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
BRDH	0000 0000	0000 0000	uuuu uuuu
BRDL	0000 0000	0000 0000	uuuu uuuu
UFCR	--00 0000	--00 0000	--uu uuuu
TXR_RXR	xxxx xxxx	xxxx xxxx	uuuu uuuu
RxCNT	---- -000	---- -000	---- -uuu
STMC0	0000 0---	0000 0---	uuuu u---
STMC1	0000 0000	0000 0000	uuuu uuuu
STMDL	0000 0000	0000 0000	uuuu uuuu
STMDH	0000 0000	0000 0000	uuuu uuuu
STMAL	0000 0000	0000 0000	uuuu uuuu
STMAH	0000 0000	0000 0000	uuuu uuuu
STMRP	0000 0000	0000 0000	uuuu uuuu
IFS	---- -000	---- -000	---- -uuu
PTM0C0	0000 0---	0000 0---	uuuu u---
PTM0C1	0000 0000	0000 0000	uuuu uuuu
PTM0DL	0000 0000	0000 0000	uuuu uuuu
PTM0DH	---- --00	---- --00	---- --uu
PTM0AL	0000 0000	0000 0000	uuuu uuuu
PTM0AH	---- --00	---- --00	---- --uu
PTM0RPL	0000 0000	0000 0000	uuuu uuuu
PTM0RPH	---- --00	---- --00	---- --uu
PTM1C0	0000 0---	0000 0---	uuuu u---
PTM1C1	0000 0000	0000 0000	uuuu uuuu
PTM1DL	0000 0000	0000 0000	uuuu uuuu
PTM1DH	---- --00	---- --00	---- --uu
PTM1AL	0000 0000	0000 0000	uuuu uuuu
PTM1AH	---- --00	---- --00	---- --uu
PTM1RPL	0000 0000	0000 0000	uuuu uuuu
PTM1RPH	---- --00	---- --00	---- --uu
PTM2C0	0000 0---	0000 0---	uuuu u---
PTM2C1	0000 0000	0000 0000	uuuu uuuu
PTM2DL	0000 0000	0000 0000	uuuu uuuu
PTM2DH	0000 0000	0000 0000	uuuu uuuu
PTM2AL	0000 0000	0000 0000	uuuu uuuu
PTM2AH	0000 0000	0000 0000	uuuu uuuu
PTM2RPL	0000 0000	0000 0000	uuuu uuuu
PTM2RPH	0000 0000	0000 0000	uuuu uuuu
PTM3C0	0000 0---	0000 0---	uuuu u---
PTM3C1	0000 0000	0000 0000	uuuu uuuu
PTM3DL	0000 0000	0000 0000	uuuu uuuu
PTM3DH	0000 0000	0000 0000	uuuu uuuu
PTM3AL	0000 0000	0000 0000	uuuu uuuu
PTM3AH	0000 0000	0000 0000	uuuu uuuu
PTM3RPL	0000 0000	0000 0000	uuuu uuuu
PTM3RPH	0000 0000	0000 0000	uuuu uuuu
EEC	0000 0000	0000 0000	uuuu uuuu

Note: “u” stands for unchanged
“x” stands for “unknown”
“-” stands for unimplemented

Input/Output Ports

Holtek microcontrollers offer considerable flexibility on their I/O ports. With the input or output designation of every pin fully under user program control, pull-high selections for all ports and wake-up selections on certain pins, the user is provided with an I/O structure to meet the needs of a wide range of application possibilities.

The device provides bidirectional input/output lines. These I/O ports are mapped to the RAM Data Memory with specific addresses as shown in the Special Purpose Data Memory table. All of these I/O ports can be used for input and output operations. For input operation, these ports are non-latching, which means the inputs must be ready at the T2 rising edge of instruction "MOV A, [m]", where m denotes the port address. For output operation, all the data is latched and remains unchanged until the output latch is rewritten.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	PB7	PB6	PB5	PB4	D3	PB2	PB1	PB0
PBC	PBC7	PBC6	PBC5	PBC4	D3	PBC2	PBC1	PBC0
PBPU	PBPU7	PBPU6	PBPU5	PBPU4	D3	PBPU2	PBPU1	PBPU0
PC	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
PCC	PCC7	PCC6	PCC5	PCC4	PCC3	PCC2	PCC1	PCC0
PCPU	PCPU7	PCPU6	PCPU5	PCPU4	PCPU3	PCPU2	PCPU1	PCPU0
PDC	—	—	—	—	—	—	PDC1	PDC0
PDPUS0	—	—	—	—	PDPUS03	PDPUS02	PDPUS01	PDPUS00

"—": Unimplemented, read as "0"

I/O Logic Function Register List

Pull-high Resistors

Many product applications require pull-high resistors for their switch inputs usually requiring the use of an external resistor. To eliminate the need for these external resistors, all I/O pins, when configured as a digital input have the capability of being connected to an internal pull-high resistor. These pull-high resistors are selected using the relevant pull-high control registers and are implemented using weak PMOS transistors. These pull-high resistors are selected using the PxPU and PDPUS0 registers, and are implemented using weak PMOS transistors.

Note that the pull-high resistor can be controlled by the relevant pull-high control register only when the pin-shared functional pin is selected as a digital input or NMOS output. Otherwise, the pull-high resistors cannot be enabled.

- **PxPU Register**

Bit	7	6	5	4	3	2	1	0
Name	PxPU7	PxPU6	PxPU5	PxPU4	PxPU3	PxPU2	PxPU1	PxPU0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

PxPUn: I/O Port x Pin pull-high function control
0: Disable
1: Enable

The PxPUn bit is used to control the pin pull-high function. Here the "x" is the Port name which can be A, B or C. However, the actual available bits for each I/O Port may be different.

• **PDPUS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PDPUS03	PDPUS02	PDPUS01	PDPUS00
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **PDPUS03~PDPUS02:** PD1 pull-high resister selection
00: Disable
01: 4kΩ
10: 10kΩ
11: 30kΩ (typ.) @ 5V / 60kΩ (typ.) @ 3V
Note that for the PD1 line which is internally connected to the PD PHY, its pull-high function can still be controlled by these bits. When the I^2C is used for commination between the MCU and PD PHY, these bits can be configured according to application requirements.

Bit 1~0 **PDPUS01~PDPUS00:** PD0 pull-high resister selection
00: Disable
01: 4kΩ
10: 10kΩ
11: 30kΩ (typ.) @ 5V / 60kΩ (typ.) @ 3V
Note that for the PD0 line which is internally connected to the PD PHY, its pull-high function can still be controlled by these bits. When the I^2C is used for commination between the MCU and PD PHY, these bits can be configured according to application requirements.

Port A Wake-up

The HALT instruction forces the microcontroller into the SLEEP or IDLE Mode which preserves power, a feature that is important for battery and other low-power applications. Various methods exist to wake up the microcontroller, one of which is to change the logic condition on one of the Port A pins from high to low. This function is especially suitable for applications that can be woken up via external switches. Each pin on Port A can be selected individually to have this wake-up feature using the PAWU register.

Note that the wake-up function can be controlled by the wake-up control register only when the pin is selected as a general purpose input and the MCU enters the IDLE or SLEEP mode.

• **PAWU Register**

Bit	7	6	5	4	3	2	1	0
Name	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **PAWU7~PAWU0:** Port A pin wake-up function control
0: Disable
1: Enable
Note that for the PA4~PA5 lines, which are internally connected to the PD PHY, the corresponding bits should be kept unchanged after power on.

I/O Port Control Registers

Each Port has its own control register which controls the input/output configuration. With this control register, each I/O pin with or without pull-high resistors can be reconfigured dynamically under software control. For the I/O pin to function as an input, the corresponding bit of the control register must be written as a "1". This will then allow the logic state of the input pin to be directly

read by instructions. When the corresponding bit of the control register is written as a "0", the I/O pin will be setup as a CMOS output. If the pin is currently setup as an output, instructions can still be used to read the output register.

However, it should be noted that the program will in fact only read the status of the output data latch and not the actual logic status of the output pin when the IECM is cleared to "0".

• **PxC Register**

Bit	7	6	5	4	3	2	1	0
Name	PxC7	PxC6	PxC5	PxC4	PxC3	PxC2	PxC1	PxC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	1	1	1	1	1

PxCn: I/O Port x Pin type selection
0: Output
1: Input

The PxCn bit is used to control the pin type selection. Here the "x" is the Port name which can be A, B, C or D. However, the actual available bits for each I/O Port may be different.

Note that for the PA4~PA5, PB1~PB2 and PD0~PD1 lines which are internally connected to the PD PHY, the corresponding bits should be properly configured after power on to implement correct interconnection. The PA4~PA5, PB1 and PD0~PD1 lines should be set as outputs and the PB2 line should be set as an input. In addition, the port control bit denoted as "D3" for port B should be cleared to 0 to set the corresponding pin as an output after power on. This can prevent the device from consuming power due to input floating states for any unbonded pins.

I/O Port Source Current Control

The device supports different output source current driving capability for each I/O port. With the selection register, SLEDCn, specific I/O port can support four levels of the source current driving capability. These source current selection bits are available when the corresponding pin is configured as a CMOS output. Otherwise, these select bits have no effect. Users should refer to the Input/Output Characteristics section to select the desired output source current for different applications.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SLEDC0	SLEDC07	SLEDC06	SLEDC05	SLEDC04	SLEDC03	SLEDC02	SLEDC01	SLEDC00
SLEDC1	—	—	—	—	SLEDC13	SLEDC12	SLEDC11	SLEDC10

I/O Port Source Current Control Register List

• **SLEDC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	SLEDC07	SLEDC06	SLEDC05	SLEDC04	SLEDC03	SLEDC02	SLEDC01	SLEDC00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **SLEDC07~SLEDC06:** PB7~PB4 source current selection
00: Source current = Level 0 (min.)
01: Source current = Level 1
10: Source current = Level 2
11: Source current = Level 3 (max.)

Note: For PB4~PB5, refer to the Input/Output (with Multi-power) D.C. Characteristics section to obtain the exact value. While for PB6~PB7, refer to the Input/Output (without Multi-power) D.C. Characteristics section to obtain the exact value.

Bit 5~4 **SLEDC05~SLEDC04:** PB0 source current selection
00: Source current = Level 0 (min.)
01: Source current = Level 1
10: Source current = Level 2
11: Source current = Level 3 (max.)

Bit 3~2 **SLEDC03~SLEDC02:** PA7~PA6 source current selection
00: Source current = Level 0 (min.)
01: Source current = Level 1
10: Source current = Level 2
11: Source current = Level 3 (max.)

Bit 1~0 **SLEDC01~SLEDC00:** PA3~PA0 source current selection
00: Source current = Level 0 (min.)
01: Source current = Level 1
10: Source current = Level 2
11: Source current = Level 3 (max.)

• **SLEDC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	SLEDC13	SLEDC12	SLEDC11	SLEDC10
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **SLEDC13~SLEDC12:** PC7~PC4 source current selection
00: Source current = Level 0 (min.)
01: Source current = Level 1
10: Source current = Level 2
11: Source current = Level 3 (max.)

Bit 1~0 **SLEDC11~SLEDC10:** PC3~PC0 source current selection
00: Source current = Level 0 (min.)
01: Source current = Level 1
10: Source current = Level 2
11: Source current = Level 3 (max.)

I/O Port Power Source Control

The device supports different I/O port power source selections for PB4~PB5 pins and PD0~PD1 lines. With the exception of $\overline{\text{RES}}$ OCDS, the multi-power function is only effective when the pin is set to have a digital input or output function.

The port power can come from either the power pin VDD or VDDIO, which is determined using the PMPS3~PMPS0 bits in the PMPS register. The VDDIO power pin function should first be selected using the corresponding pin-shared function selection bits if the port power is supposed to come from the VDDIO pin.

An important point to know is that the input power voltage on the VDDIO pin should be equal to or less than the device supply power voltage V_{DD} when the VDDIO pin is selected as the port power supply pin.

• **PMPS Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PMPS3	PMPS2	PMPS1	PMPS0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **PMPS3~PMPS2:** PD0~PD1 line power supply selection
0x: V_{DD}
1x: V_{DDIO}
Note that for the PD0~PD1 lines which are internally connected to the PD PHY, their multi-power function can still be controlled by these bits. When the I^2C is used for commination between the MCU and PD PHY, these bits can be configured according to application requirements.
If the PC2 pin is switched to the VDDIO function, and the PMPS3~PMPS2 bits are set to "1x", the VDDIO pin input voltage can be used for PD0~PD1 line power.

Bit 1~0 **PMPS1~PMPS0:** PB4~PB5 pin power supply selection
0x: V_{DD}
1x: V_{DDIO}
If the PC2 pin is switched to the VDDIO function, and the PMPS1~PMPS0 bits are set to "1x", the VDDIO pin input voltage can be used for PB4~PB5 pin power.

Pin-shared Functions

The flexibility of the microcontroller range is greatly enhanced by the use of pins that have more than one function. Limited numbers of pins can force serious design constraints on designers but by supplying pins with multi-functions, many of these difficulties can be overcome. For these pins, the desired function of the multi-function I/O pins is selected by a series of registers via the application program control.

Pin-shared Function Selection Registers

The limited number of supplied pins in a package can impose restrictions on the amount of functions a certain device can contain. However by allowing the same pins to share several different functions and providing a means of function selection, a wide range of different functions can be incorporated into even relatively small package sizes. The device includes Port "x" output function Selection register "n", labeled as PxSn, and Input Function Selection register, labeled as IFS, which can select the desired functions of the multi-function pin-shared pins.

The most important point to note is to make sure that the desired pin-shared function is properly selected and also deselected. For most pin-shared functions, to select the desired pin-shared function, the pin-shared function should first be correctly selected using the corresponding pin-shared control register. After that the corresponding peripheral functional setting should be configured and then the peripheral function can be enabled. However, a special point must be noted for some digital input pins, such as INTn, xTCKn, xTPnI, etc., which share the same pin-shared control configuration with their corresponding general purpose I/O functions when setting the relevant pin-shared control bit fields. To select these pin functions, in addition to the necessary pin-shared control and peripheral functional setup aforementioned, they must also be setup as input by setting the corresponding bit in the I/O port control register. To correctly deselect the pin-shared function, the peripheral function should first be disabled and then the corresponding pin-shared function control register can be modified to select other pin-shared functions.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PAS0	PAS07	PAS06	—	—	PAS03	PAS02	—	—
PAS1	PAS17	PAS16	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10
PBS0	—	—	—	—	—	—	PBS01	PBS00
PBS1	PBS17	PBS16	PBS15	PBS14	PBS13	PBS12	PBS11	PBS10
PCS0	PCS07	PCS06	PCS05	PCS04	PCS03	PCS02	PCS01	PCS00
PCS1	PCS17	PCS16	PCS15	PCS14	PCS13	PCS12	PCS11	PCS10
PDS0	—	—	—	—	PDS03	PDS02	PDS01	PDS00
IFS	—	—	—	—	—	SCLMPS	SDAMPS	RXPS

Pin-shared Function Selection Register List

- **PAS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PAS07	PAS06	—	—	PAS03	PAS02	—	—
R/W	R/W	R/W	—	—	R/W	R/W	—	—
POR	0	0	—	—	0	0	—	—

Bit 7~6 **PAS07~PAS06:** PA3 pin-shared function selection
00: PA3
01: OPINN0
10: OPINP
11: PA3

Bit 5~4 Unimplemented, read as "0"

Bit 3~2 **PAS03~PAS02:** PA1 pin-shared function selection
00: PA1
01: SCK/SCL
10: OPARO
11: PA1

Bit 1~0 Unimplemented, read as "0"

- **PAS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PAS17	PAS16	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PAS17~PAS16:** PA7 pin-shared function selection
00: PA7
01: SDO
10: PTP0
11: AN1

Bit 5~4 **PAS15~PAS14:** PA6 pin-shared function selection
00: PA6
01: SDI/SDA
10: PTP1
11: AN0

Bit 3~2 **PAS13~PAS12:** Reserved
Note that these bits must be kept unchanged after power on.

Bit 1~0 **PAS11~PAS10:** Reserved
Note that these bits must be kept unchanged after power on.

• **PBS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PBS01	PBS00
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **PBS01~PBS00:** PB0 pin-shared function selection
00: PB0/PTCK0
01: $\overline{SCS}$
10: PTP1B
11: PB0/PTCK0

• **PBS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PBS17	PBS16	PBS15	PBS14	PBS13	PBS12	PBS11	PBS10
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PBS17~PBS16:** PB7 pin-shared function selection
00: PB7
01: D1-
10: AN3
11: PB7

Bit 5~4 **PBS15~PBS14:** PB6 pin-shared function selection
00: PB6
01: D1+
10: AN2
11: PB6

Bit 3~2 **PBS13~PBS12:** PB5 pin-shared function selection
00: PB5
01: TX
10: PB5
11: PB5

Bit 1~0 **PBS11~PBS10:** PB4 pin-shared function selection
00: PB4
01: RX/TX
10: PB4
11: PB4

• **PCS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PCS07	PCS06	PCS05	PCS04	PCS03	PCS02	PCS01	PCS00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PCS07~PCS06:** PC3 pin-shared function selection
00: PC3/STCK
01: STP
10: AN6
11: VREF

Bit 5~4 **PCS05~PCS04:** PC2 pin-shared function selection
00: PC2/STPI
01: VDDIO
10: STPB
11: PC2/STPI

Bit 3~2 **PCS03~PCS02:** PC1 pin-shared function selection
00: PC1
01: D2-
10: AN5
11: PC1

Bit 1~0 **PCS01~PCS00:** PC0 pin-shared function selection
00: PC0
01: D2+
10: AN4
11: PC0

• PCS1 Register

Bit	7	6	5	4	3	2	1	0
Name	PCS17	PCS16	PCS15	PCS14	PCS13	PCS12	PCS11	PCS10
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PCS17~PCS16:** PC7 pin-shared function selection
00: PC7/PTP3I
01: PTP2
10: SDAM
11: AN10

Bit 5~4 **PCS15~PCS14:** PC6 pin-shared function selection
00: PC6/INT1/PTCK3
01: PTP2B
10: SCLM
11: AN9

Bit 3~2 **PCS13~PCS12:** PC5 pin-shared function selection
00: PC5/PTP2I
01: PTP3
10: AN8
11: PC5/PTP2I

Bit 1~0 **PCS11~PCS10:** PC4 pin-shared function selection
00: PC4/INT0/PTCK2
01: PTP3B
10: AN7
11: VREFI

• PDS0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PDS03	PDS02	PDS01	PDS00
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as “0”

Bit 3~2 **PDS03~PDS02:** Reserved
Note that these bits must be set to “01” when the I^2C is used for commination between the MCU and PD PHY. Otherwise, these bits should be fixed at “00”.

Bit 1~0 **PDS01~PDS00:** Reserved
Note that these bits must be set to “01” when the I^2C is used for commination between the MCU and PD PHY. Otherwise, these bits should be fixed at “00”.

• **IFS Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	SCLMPS	SDAMPS	RXPS
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3 Unimplemented, read as "0"

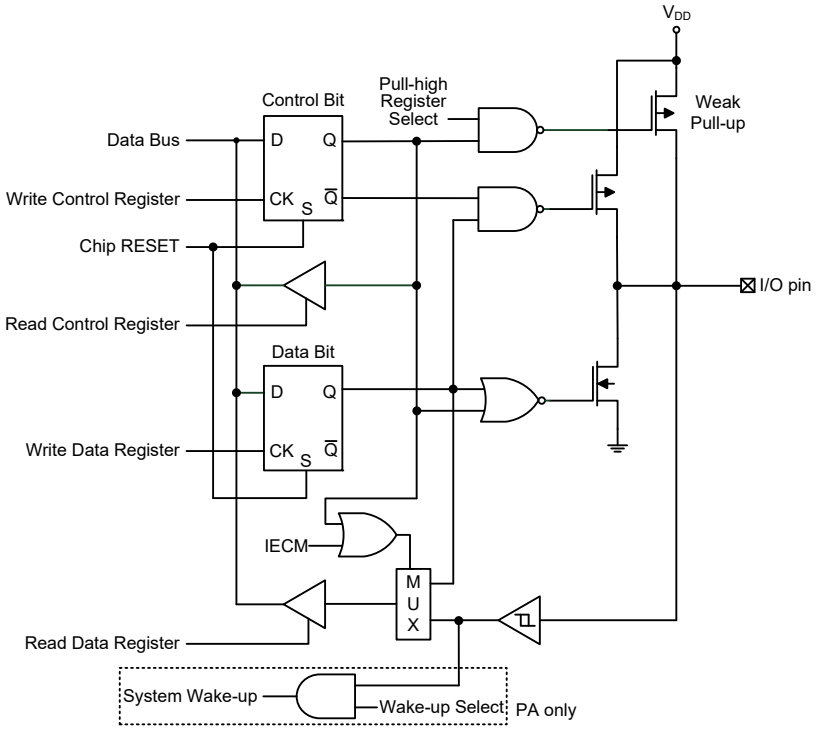
Bit 2 **SCLMPS:** SCLM source selection
0: PD0 line
1: PC6
Note that for the PD0 line which is internally connected to the PD PHY, this bit must be fixed at "0" when the I²C is used for commination between the MCU and PD PHY. Otherwise, this bit should be set to "1".

Bit 1 **SDAMPS:** SDAM source selection
0: PD1 line
1: PC7
Note that for the PD1 line which is internally connected to the PD PHY, this bit must be fixed at "0" when the I²C is used for commination between the MCU and PD PHY. Otherwise, this bit should be set to "1".

Bit 0 **RXPS:** RX/TX input source pin selection
0: PB4
1: Reserved

I/O Pin Structures

The accompanying diagram illustrates the internal structures of the I/O logic function. As the exact logical construction of the I/O pin will differ from this drawing, it is supplied as a guide only to assist with the functional understanding of the logic function I/O pins. The wide range of pin-shared structures does not permit all types to be shown.



Logic Function Input/Output Structure

READ PORT Function

The READ PORT function is used to manage the reading of the output data from the data latch or I/O pin, which is specially designed for the IEC 60730 self-diagnostic test on the I/O function and A/D paths. There is a register, IECC, which is used to control the READ PORT function. If the READ PORT function is disabled, the pin function will operate as the selected pin-shared function. When a specific data pattern, "11001010", is written into the IECC register, the internal signal named IECM will be set high to enable the READ PORT function. If the READ PORT function is enabled, the value on the corresponding pins will be passed to the accumulator ACC when the read port instruction "mov a, Px" is executed where the "x" stands for the corresponding I/O port name.

Note that the READ PORT mode can only control the input path and will not affect the pin-shared function assignment and the current MCU operation. However, when the IECC register content is set to any other values rather than "11001010", the IECM internal signal will be cleared to 0 to disable the READ PORT function, and the reading path will be set from the latch. If the READ PORT function is disabled, the pin function will operate as the selected pin-shared function.

- **IECC Register**

Bit	7	6	5	4	3	2	1	0
Name	IECS7	IECS6	IECS5	IECS4	IECS3	IECS2	IECS1	IECS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

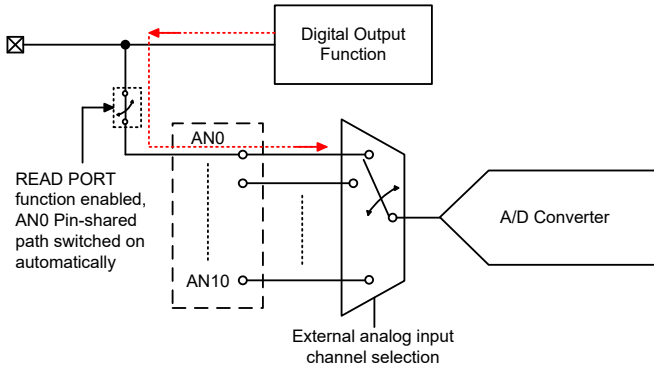
Bit 7~0 **IECS7~IECS0:** READ PORT function enable control bit 7 ~ bit 0
11001010: IECM=1 – READ PORT function is enabled
Others: IECM=0 – READ PORT function is disabled

READ PORT Function	Disabled		Enabled	
Port Control Register Bit – PxC.n	1	0	1	0
I/O Function	Pin value	Data latch value	Pin value	
Digital Input Function				
Digital Output Function (except I²C, SIM and UART)	0			
I²C: SDAM, SCLM SIM: SCK/SCL, SDI/SDA UART: RX/TX	Pin value			
Analog Function	0			

Note: The value on the above table is the content of the ACC register after "mov a, Px" instruction is executed where "x" means the relevant port name.

The additional function of the READ PORT mode is to check the A/D path. When the READ PORT function is disabled, the A/D path from the external pin to the internal analog input will be switched off if the A/D input pin function is not selected by the corresponding selection bits. For the MCU with A/D converter channels, such as A/D AN0~AN10, the desired A/D channel can be switched on by properly configuring the external analog input channel selection bits in the A/D Control Register together with the corresponding analog input pin function is selected. However, the additional function of the READ PORT mode is to force the A/D path to be switched on. For example, when the AN0 is selected as the analog input channel as the READ PORT function is enabled, the AN0 analog input path will be switched on even if the AN0 analog input pin function is not selected. In this way, the AN0 analog input path can be examined by internally connecting the digital output on this shared pin with the AN0 analog input pin switch and then converting the corresponding digital data without any external analog input voltage connected.

Note that the A/D converter reference voltage should be equal to the I/O power supply voltage when examining the A/D path using the READ PORT function.



A/D Channel Input Path Internally Connection

Programming Considerations

Within the user program, one of the things first to consider is port initialisation. After a reset, all of the I/O data and port control registers will be set high. This means that all I/O pins will be defaulted to an input state, the level of which depends on the other connected circuitry and whether pull-high selections have been chosen. If the port control registers are then programmed to setup some pins as outputs, these output pins will have an initial high output value unless the associated port data registers are first programmed. Selecting which pins are inputs and which are outputs can be achieved byte-wide by loading the correct values into the appropriate port control register or by programming individual bits in the port control register using the "SET [m].i" and "CLR [m].i" instructions. Note that when using these bit control instructions, a read-modify-write operation takes place. The microcontroller must first read in the data on the entire port, modify it to the required new bit values and then rewrite this data back to the output ports.

Port A has the additional capability of providing wake-up functions. When the device is in the SLEEP or IDLE Mode, various methods are available to wake the device up. One of these is a high to low transition of any of the Port A pins. Single or multiple pins on Port A can be setup to have this function.

Timer Modules – TM

One of the most fundamental functions in any microcontroller device is the ability to control and measure time. To implement time related functions the device includes several Timer Modules, generally abbreviated to the name TM. The TMs are multi-purpose timing units and serve to provide operations such as Timer/Counter, Input Capture, Compare Match Output and Single Pulse Output as well as being the functional unit for the generation of PWM signals. Each of the TMs has two interrupts. The addition of input and output pins for each TM ensures that users are provided with timing units with a wide and flexible range of features.

The common features of the different TM types are described here with more detailed information provided in the individual Standard and Periodic TM sections.

Introduction

The device contains several TMs and each individual TM can be categorised as a certain type, namely Standard Type TM or Periodic Type TM. Although similar in nature, the different TM types vary in their feature complexity. The common features to all of the Standard and Periodic TMs will be described in this section and the detailed operation regarding each of the TM types will be described in separate sections. The main features and differences between the two types of TMs are summarised in the accompanying table.

TM Function	STM	PTM
Timer/Counter	√	√
Input Capture	√	√
Compare Match Output	√	√
PWM Output	√	√
Single Pulse Output	√	√
PWM Alignment	Edge	Edge
PWM Adjustment Period & Duty	Duty or Period	Duty or Period

TM Function Summary

TM Operation

The different types of TM offer a diverse range of functions, from simple timing operations to PWM signal generation. The key to understanding how the TM operates is to see it in terms of a free running count-up counter whose value is then compared with the value of pre-programmed internal comparators. When the free running count-up counter has the same value as the pre-programmed comparator, known as a compare match situation, a TM interrupt signal will be generated which can clear the counter and perhaps also change the condition of the TM output pin. The internal TM counter is driven by a user selectable clock source, which can be an internal clock or an external pin.

TM Clock Source

The clock source which drives the main counter in each TM can originate from various sources. The selection of the required clock source is implemented using the xTnCK2~xTnCK0 bits in the xTMn control registers, where “x” stands for S or P type TM and “n” stands for the specific TM serial number. For STM there is no serial number “n” in the relevant pins, registers and control bits since there is only one STM in the device. The clock source can be a ratio of the system clock, f_{SYS}, or the internal high clock, f_H, the f_{SUB} clock source or the external xTCKn pin. The xTCKn pin clock source is used to allow an external signal to drive the TM as an external clock source for event counting.

TM Interrupts

The Standard or Periodic type TM has two internal interrupt, one for each of the internal comparator A or comparator P, which generate a TM interrupt when a compare match condition occurs. When a TM interrupt is generated, it can be used to clear the counter and also to change the state of the TM output pin.

TM External Pins

Each of the TMs, irrespective of what type, has one or two TM input pins, with the label xTCKn and xTPnI respectively, except that the PTM1 has no external input pins. The xTMn input pin, xTCKn, is essentially a clock source for the xTMn and is selected using the xTnCK2~xTnCK0 bits in the xTMnC0 register. This external TM input pin allows an external clock source to drive the internal TM. The xTCKn input pin can be chosen to have either a rising or falling active edge. The xTCKn pins are also used as the external trigger input pin in single pulse output mode for the xTMn.

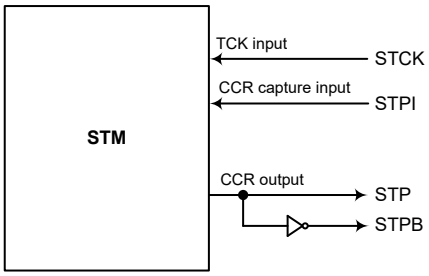
The other xTM input pin, xTPnI, is the capture input whose active edge can be a rising edge, a falling edge or both rising and falling edges and the active edge transition type is selected using the xTnIO1~xTnIO0 bits in the xTMnC1 register. There is another capture input, PTCKn, for PTMn capture input mode, which can be used as the external trigger input source except the PTPnI pin.

The TMs each have one or two output pins, xTPn and xTPnB. The xTPnB pin outputs the inverted signal of the xTPn. When the TM is in the Compare Match Output Mode, these pins can be controlled by the TM to switch to a high or low level or to toggle when a compare match situation occurs. The external xTPn and xTPnB output pin is also the pin where the TM generates the PWM output waveform.

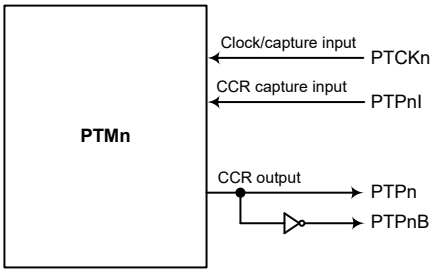
As the TM input and output pins are pin-shared with other functions, the TM input and output functions must first be setup using relevant pin-shared function selection register. The details of the pin-shared function selection are described in the pin-shared function section.

STM		PTM	
Input	Output	Input	Output
STCK, STPI	STP, STPB	PTCK0 — PTCK2, PTP2I PTCK3, PTP3I	PTP0 PTP1, PTP1B PTP2, PTP2B PTP3, PTP3B

TM External Pins



STM Function Pin Block Diagram



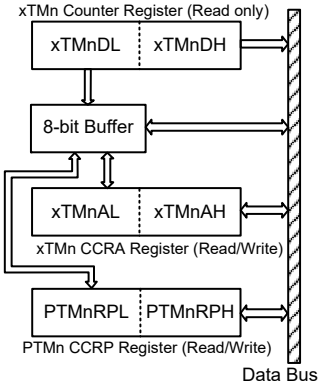
Note: PTP0I, PTP0B, PTCK1 and PTP1I are not externally bounded.

PTM Function Pin Block Diagram (n=0~3)

Programming Considerations

The TM Counter Registers and the Capture/Compare CCRA and CCRP registers, all have a low and high byte structure. The high bytes can be directly accessed, but as the low bytes can only be accessed via an internal 8-bit buffer, reading or writing to these register pairs must be carried out in a specific way. The important point to note is that data transfer to and from the 8-bit buffer and its related low byte only takes place when a write or read operation to its corresponding high byte is executed.

As the CCRA and CCRP registers are implemented in the way shown in the following diagram and accessing these register pairs is carried out in a specific way as described above, it is recommended to use the "MOV" instruction to access the CCRA and CCRP low byte registers, named xTMnAL and PTMnRPL, using the following access procedures. Accessing the CCRA or CCRP low byte registers without following these access procedures will result in unpredictable values.

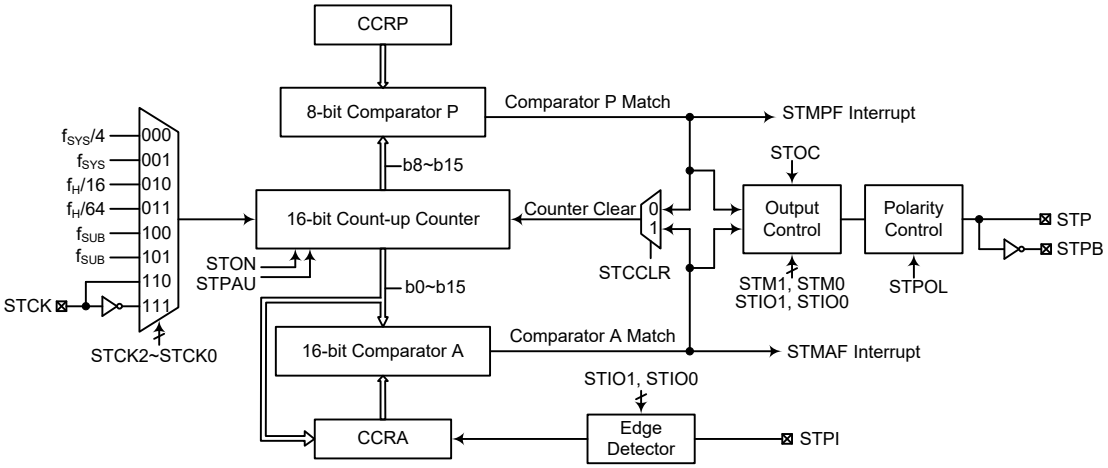


The following steps show the read and write procedures:

- Writing Data to CCRA or CCRP
  - Step 1. Write data to Low Byte xTMnAL or PTMnRPL
    - Note that here data is only written to the 8-bit buffer.
  - Step 2. Write data to High Byte xTMnAH or PTMnRPH
    - Here data is written directly to the high byte registers and simultaneously data is latched from the 8-bit buffer to the Low Byte registers.
- Reading Data from the Counter Registers and CCRA or CCRP
  - Step 1. Read data from the High Byte xTMnDH, xTMnAH or PTMnRPH
    - Here data is read directly from the High Byte registers and simultaneously data is latched from the Low Byte register into the 8-bit buffer.
  - Step 2. Read data from the Low Byte xTMnDL, xTMnAL or PTMnRPL
    - This step reads data from the 8-bit buffer.

Standard Type TM – STM

The Standard Type TM contains five operating modes, which are Compare Match Output, Timer/Event Counter, Capture Input, Single Pulse Output and PWM Output modes. The Standard TM can be controlled with two external input pins and can drive two external output pins.



Note: The STM external pins are pin-shared with other functions, so before using the STM function, the pin-shared function registers must be set properly to enable the STM pin function. The STCK and STPI pins, if used, must also be set as an input by setting the corresponding bits in the port control register.

Standard Type TM Block Diagram

Standard TM Operation

The Standard Type TM core is a 16-bit count-up counter which is driven by a user selectable internal or external clock source. There are also two internal comparators with the names, Comparator A and Comparator P. These comparators will compare the value in the counter with CCRP and CCRA registers. The CCRP comparator is 8-bit wide whose value is compared with the highest 8 bits in the counter while the CCRA is 16 bits and therefore compares all counter bits.

The only way of changing the value of the 16-bit counter using the application program, is to clear the counter by changing the STON bit from low to high. The counter will also be cleared automatically by a counter overflow or a compare match with one of its associated comparators. When these conditions occur, a STM interrupt signal will also usually be generated. The Standard Type TM can operate in a number of different operational modes, can be driven by different clock sources including an input pin and can also control two output pins. All operating setup conditions are selected using relevant internal registers.

Standard Type TM Register Description

Overall operation of the Standard TM is controlled using a series of registers. A read only register pair exists to store the internal counter 16-bit value, while a read/write register pair exists to store the internal 16-bit CCRA value. The STMRP register is used to store the 8-bit CCRP bits. The remaining two registers are control registers which setup the different operating and control modes.

Register Name	Bit							
	7	6	5	4	3	2	1	0
STMC0	STPAU	STCK2	STCK1	STCK0	STON	—	—	—
STMC1	STM1	STM0	STIO1	STIO0	STOC	STPOL	STDPX	STCCLR
STMDL	D7	D6	D5	D4	D3	D2	D1	D0
STMDH	D15	D14	D13	D12	D11	D10	D9	D8

Register Name	Bit							
	7	6	5	4	3	2	1	0
STMAL	D7	D6	D5	D4	D3	D2	D1	D0
STMAH	D15	D14	D13	D12	D11	D10	D9	D8
STMRP	D7	D6	D5	D4	D3	D2	D1	D0

16-bit Standard TM Register List

- **STMC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	STPAU	STCK2	STCK1	STCK0	STON	—	—	—
R/W	R/W	R/W	R/W	R/W	R/W	—	—	—
POR	0	0	0	0	0	—	—	—

Bit 7 **STPAU:** STM Counter Pause Control
0: Run
1: Pause
The counter can be paused by setting this bit high. Clearing the bit to zero restores normal counter operation. When in a Pause condition the STM will remain powered up and continue to consume power. The counter will retain its residual value when this bit changes from low to high and resume counting from this value when the bit changes to a low value again.

Bit 6~4 **STCK2~STCK0:** Select STM Counter Clock
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{SUB}
101: f_{SUB}
110: STCK rising edge clock
111: STCK falling edge clock
These three bits are used to select the clock source for the STM. The external pin clock source can be chosen to be active on the rising or falling edge. The clock source f_{SYS} is the system clock, while f_H and f_{SUB} are other internal clocks, the details of which can be found in the oscillator section.

Bit 3 **STON:** STM Counter On/Off Control
0: Off
1: On
This bit controls the overall on/off function of the STM. Setting the bit high enables the counter to run while clearing the bit disables the STM. Clearing this bit to zero will stop the counter from counting and turn off the STM which will reduce its power consumption. When the bit changes state from low to high the internal counter value will be reset to zero, however when the bit changes from high to low, the internal counter will retain its residual value until the bit returns high again. If the STM is in the Compare Match Output Mode, PWM Output Mode or Single Pulse Output Mode then the STM output pin will be reset to its initial condition, as specified by the STOC bit, when the STON bit changes from low to high.

Bit 2~0 Unimplemented, read as “0”

• **STMC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	STM1	STM0	STIO1	STIO0	STOC	STPOL	STDPX	STCCLR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **STM1~STM0:** Select STM Operating Mode
00: Compare Match Output Mode
01: Capture Input Mode
10: PWM Output Mode or Single Pulse Output Mode
11: Timer/Counter Mode

These bits setup the required operating mode for the STM. To ensure reliable operation the STM should be switched off before any changes are made to the STM1 and STM0 bits. In the Timer/Counter Mode, the STM output pin state is undefined.

Bit 5~4 **STIO1~STIO0:** Select STM External Pin Function

Compare Match Output Mode
00: No change
01: Output low
10: Output high
11: Toggle output

PWM Output Mode/Single Pulse Output Mode
00: PWM output inactive state
01: PWM output active state
10: PWM output
11: Single Pulse Output

Capture Input Mode
00: Input capture at rising edge of STPI
01: Input capture at falling edge of STPI
10: Input capture at rising/falling edge of STPI
11: Input capture disabled

Timer/Counter Mode
Unused

These two bits are used to determine how the STM external pin changes state when a certain condition is reached. The function that these bits select depends upon in which mode the STM is running.

In the Compare Match Output Mode, the STIO1 and STIO0 bits determine how the STM output pin changes state when a compare match occurs from the Comparator A. The STM output pin can be setup to switch high, switch low or to toggle its present state when a compare match occurs from the Comparator A. When the bits are both zero, then no change will take place on the output. The initial value of the STM output pin should be setup using the STOC bit in the STMC1 register. Note that the output level requested by the STIO1 and STIO0 bits must be different from the initial value setup using the STOC bit otherwise no change will occur on the STM output pin when a compare match occurs. After the STM output pin changes state, it can be reset to its initial level by changing the level of the STON bit from low to high.

In the PWM Output Mode, the STIO1 and STIO0 bits determine how the STM output pin changes state when a certain compare match condition occurs. The PWM output function is modified by changing these two bits. It is necessary to only change the values of the STIO1 and STIO0 bits only after the STM has been switched off. Unpredictable PWM outputs will occur if the STIO1 and STIO0 bits are changed when the STM is running.

Bit 3 **STOC:** STM STP Output Control

Compare Match Output Mode
0: Initial low
1: Initial high

PWM Output Mode/Single Pulse Output Mode
0: Active low
1: Active high

This is the output control bit for the STM output pin. Its operation depends upon whether STM is being used in the Compare Match Output Mode or in the PWM Output Mode/Single Pulse Output Mode. It has no effect if the STM is in the Timer/Counter Mode. In the Compare Match Output Mode it determines the logic level of the STM output pin before a compare match occurs. In the PWM Output Mode it determines if the PWM signal is active high or active low. In the Single Pulse Output Mode it determines the logic level of the STM output pin when the STON bit changes from low to high.

Bit 2 **STPOL:** STM STP Output Polarity Control
0: Non-inverted
1: Inverted

This bit controls the polarity of the STP output pin. When the bit is set high the STM output pin will be inverted and not inverted when the bit is zero. It has no effect if the STM is in the Timer/Counter Mode.

Bit 1 **STDPX:** STM PWM Duty/Period Control
0: CCRP – period; CCRA – duty
1: CCRP – duty; CCRA – period

This bit determines which of the CCRA and CCRP registers are used for period and duty control of the PWM waveform.

Bit 0 **STCCLR:** STM Counter Clear Condition Selection
0: Comparator P match
1: Comparator A match

This bit is used to select the method which clears the counter. Remember that the Standard TM contains two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the STCCLR bit set high, the counter will be cleared when a compare match occurs from the Comparator A. When the bit is low, the counter will be cleared when a compare match occurs from the Comparator P or with a counter overflow. A counter overflow clearing method can only be implemented if the CCRP bits are all cleared to zero. The STCCLR bit is not used in the PWM Output, Single Pulse Output or Capture Input Mode.

• STMDL Register

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0:** STM Counter Low Byte Register bit 7 ~ bit 0
STM 16-bit Counter bit 7 ~ bit 0

• STMDH Register

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D15~D8:** STM Counter High Byte Register bit 7 ~ bit 0
STM 16-bit Counter bit 15 ~ bit 8

• **STMAL Register**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0:** STM CCRA Low Byte Register bit 7 ~ bit 0
STM 16-bit CCRA bit 7 ~ bit 0

• **STMAH Register**

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D15~D8:** STM CCRA High Byte Register bit 7 ~ bit 0
STM 16-bit CCRA bit 15 ~ bit 8

• **STMRP Register**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0:** STM CCRP 8-bit Register, Compared with the STM Counter bit 15 ~ bit 8
Comparator P Match period =
0: 65536 STM clocks
1~255: (1~255)×256 STM clocks
These eight bits are used to setup the value on the internal CCRP 8-bit register, which are then compared with the internal counter's highest eight bits. The result of this comparison can be selected to clear the internal counter if the STCCLR bit is set to zero. Setting the STCCLR bit to zero ensures that a compare match with the CCRP values will reset the internal counter. As the CCRP bits are only compared with the highest eight counter bits, the compare values exist in 256 clock cycle multiples. Clearing all eight bits to zero is in effect allowing the counter to overflow at its maximum value.

Standard Type TM Operation Modes

The Standard Type TM can operate in one of five operating modes, Compare Match Output Mode, PWM Output Mode, Single Pulse Output Mode, Capture Input Mode or Timer/Counter Mode. The operating mode is selected using the STM1 and STM0 bits in the STMC1 register.

Compare Match Output Mode

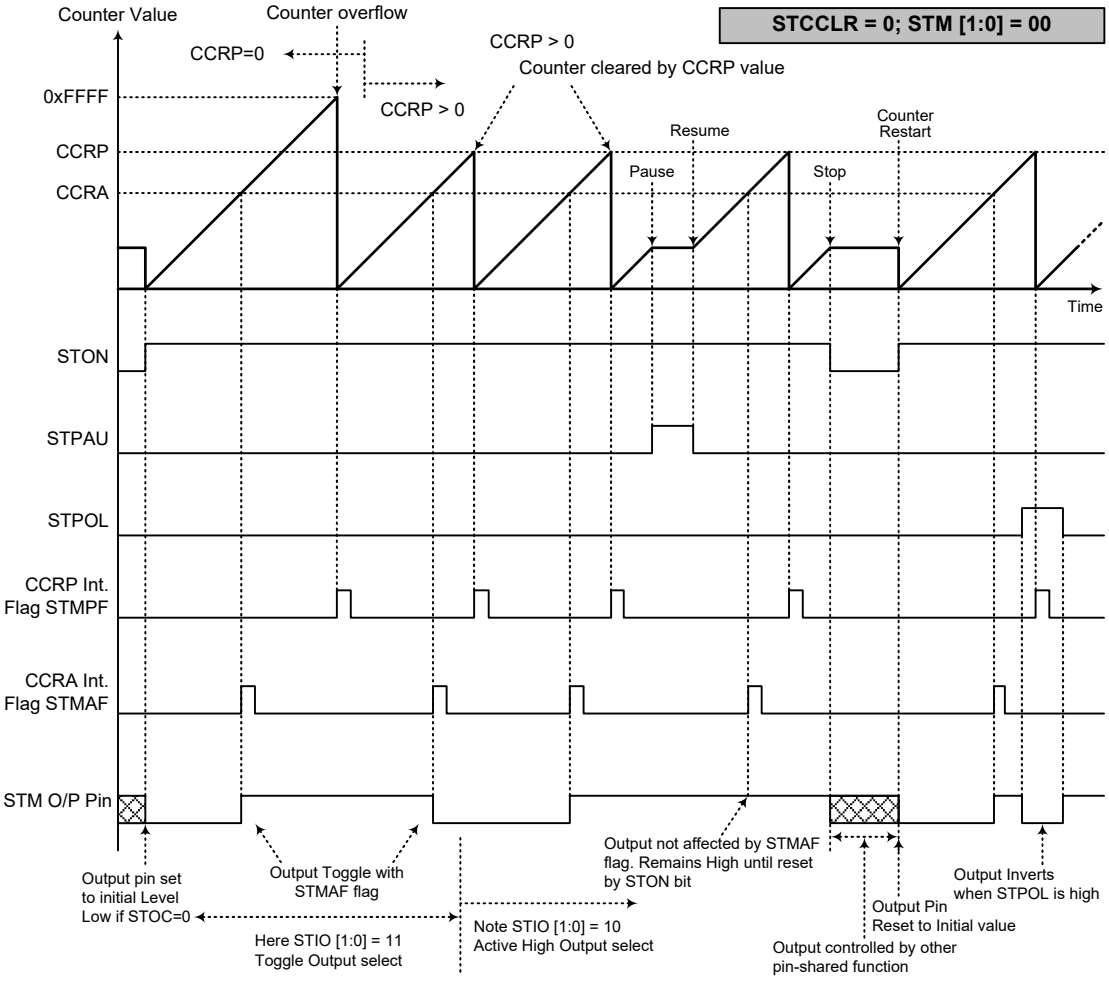
To select this mode, bits STM1 and STM0 in the STMC1 register, should be set to 00 respectively. In this mode once the counter is enabled and running it can be cleared by three methods. These are a counter overflow, a compare match from Comparator A and a compare match from Comparator P. When the STCCLR bit is low, there are two ways in which the counter can be cleared. One is when a compare match from Comparator P, the other is when the CCRP bits are all zero which allows the counter to overflow. Here both STMAF and STMPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

If the STCCLR bit in the STMC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the STMAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when

STCCLR is high no STMPF interrupt request flag will be generated. In the Compare Match Output Mode, the CCRA cannot be cleared to "0".

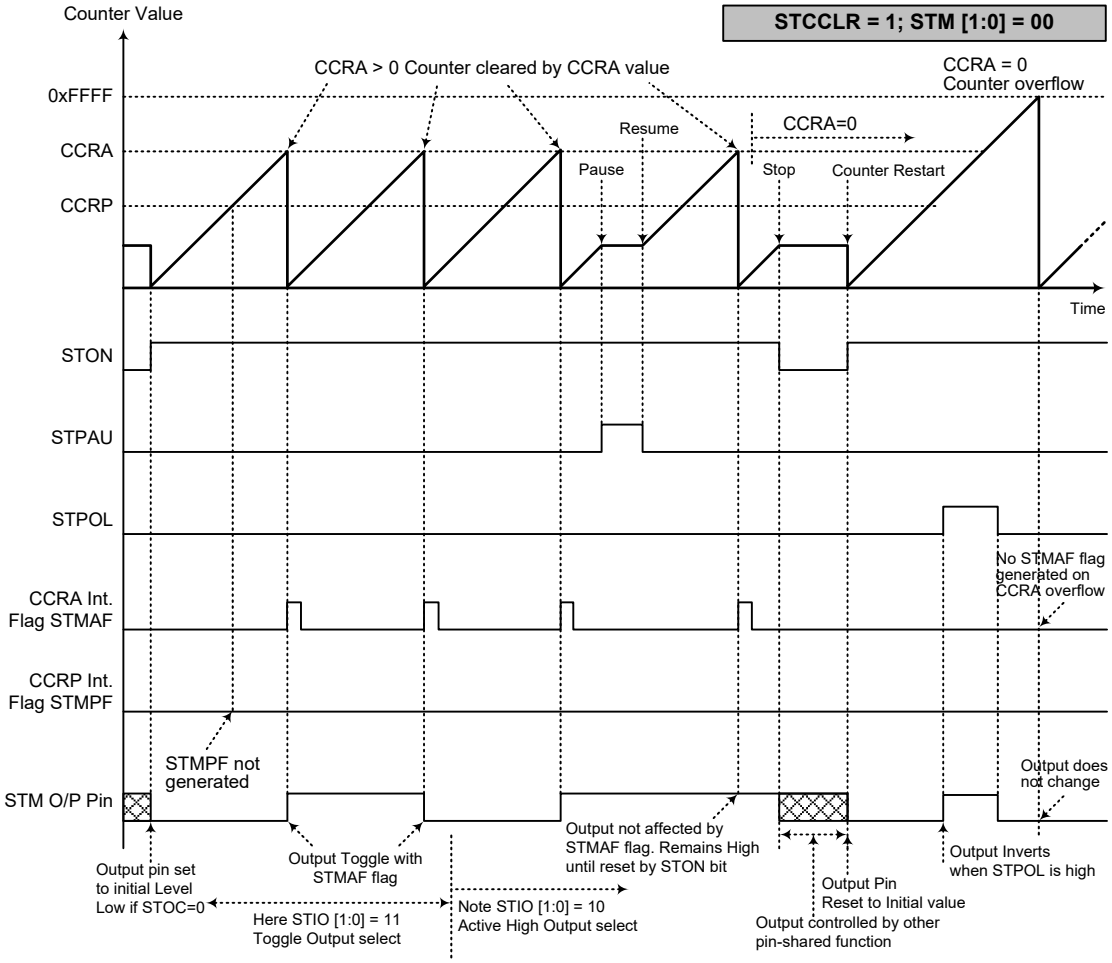
If the CCRA bits are all zero, the counter will overflow when it reaches its maximum 16-bit, FFFF Hex, value, however here the STMAF interrupt request flag will not be generated.

As the name of the mode suggests, after a comparison is made, the STM output pin, will change state. The STM output pin condition however only changes state when a STMAF interrupt request flag is generated after a compare match occurs from Comparator A. The STMPF interrupt request flag, generated from a compare match occurs from Comparator P, will have no effect on the STM output pin. The way in which the STM output pin changes state are determined by the condition of the STIO1 and STIO0 bits in the STMC1 register. The STM output pin can be selected using the STIO1 and STIO0 bits to go high, to go low or to toggle from its present condition when a compare match occurs from Comparator A. The initial condition of the STM output pin, which is setup after the STON bit changes from low to high, is setup using the STOC bit. Note that if the STIO1 and STIO0 bits are zero then no pin change will take place.



Compare Match Output Mode – STCCLR=0

Note: 1. With STCCLR=0 a Comparator P match will clear the counter
2. The STM output pin is controlled only by the STMAF flag
3. The output pin is reset to initial state by a STON bit rising edge



Compare Match Output Mode – STCCLR=1

Note: 1. With STCCLR=1 a Comparator A match will clear the counter

2. The STM output pin is controlled only by the STMAF flag
3. The output pin is reset to its initial state by a STON bit rising edge
4. A STMPF flag is not generated when STCCLR=1

Timer/Counter Mode

To select this mode, bits STM1 and STM0 in the STMC1 register should be set to 11 respectively. The Timer/Counter Mode operates in an identical way to the Compare Match Output Mode generating the same interrupt flags. The exception is that in the Timer/Counter Mode the STM output pin is not used. Therefore the above description and Timing Diagrams for the Compare Match Output Mode can be used to understand its function. As the STM output pin is not used in this mode, the pin can be used as a normal I/O pin or other pin-shared function.

PWM Output Mode

To select this mode, bits STM1 and STM0 in the STMC1 register should be set to 10 respectively and also the STIO1 and STIO0 bits should be set to 10 respectively. The PWM function within the STM is useful for applications which require functions such as motor control, heating control, illumination control etc. By providing a signal of fixed frequency but of varying duty cycle on the STM output pin, a square waveform AC waveform can be generated with varying equivalent DC RMS values.

As both the period and duty cycle of the PWM waveform can be controlled, the choice of generated waveform is extremely flexible. In the PWM Output mode, the STCCLR bit has no effect as the PWM period. Both of the CCRP and CCRA registers are used to generate the PWM waveform, the CCRP register is used to clear the internal counter and thus control the PWM waveform frequency, while the CCRA register is used to control the duty cycle. Which register is used to control either frequency or duty cycle is determined using the STDPX bit in the STMC1 register. The PWM waveform frequency and duty cycle can therefore be controlled by the values in the CCRA and CCRP registers.

An interrupt flag, one for each of the CCRA and CCRP, will be generated when a compare match occurs from either Comparator A or Comparator P. The STOC bit in the STMC1 register is used to select the required polarity of the PWM waveform while the two STIO1 and STIO0 bits are used to enable the PWM output or to force the STM output pin to a fixed high or low level. The STPOL bit is used to reverse the polarity of the PWM output waveform.

- **16-bit STM, PWM Output Mode, Edge-aligned Mode, STDPX=0**

CCRP	1~255	0
Period	CCRP×256	65536
Duty	CCRA	

If f_{SYS}=16MHz, STM clock source is f_{SYS}/4, CCRP=2 and CCRA=128,

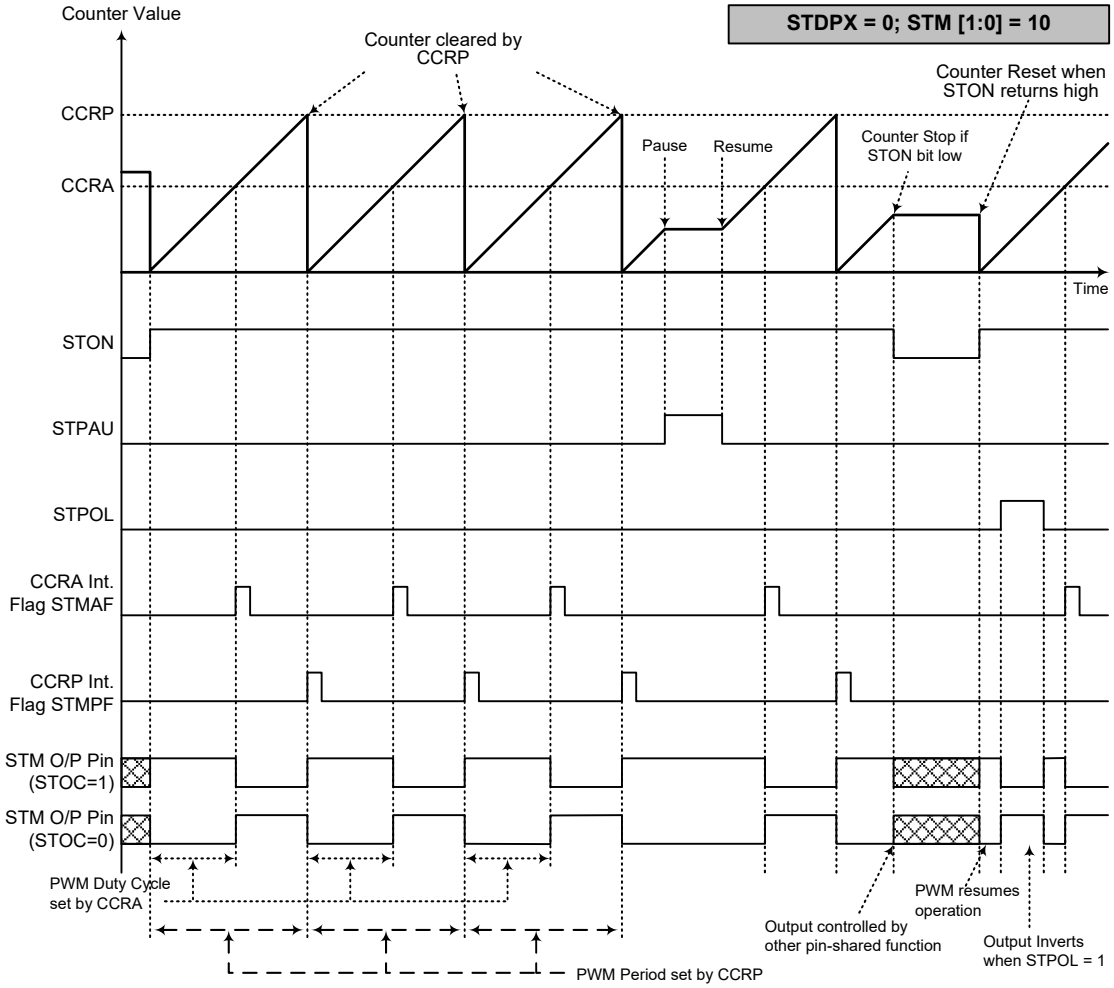
The STM PWM output frequency = (f_{SYS}/4)/(2×256)=f_{SYS}/2048=8kHz, duty=128/(2×256)=25%.

If the Duty value defined by the CCRA register is equal to or greater than the Period value, then the PWM output duty is 100%.

- **16-bit STM, PWM Output Mode, Edge-aligned Mode, STDPX=1**

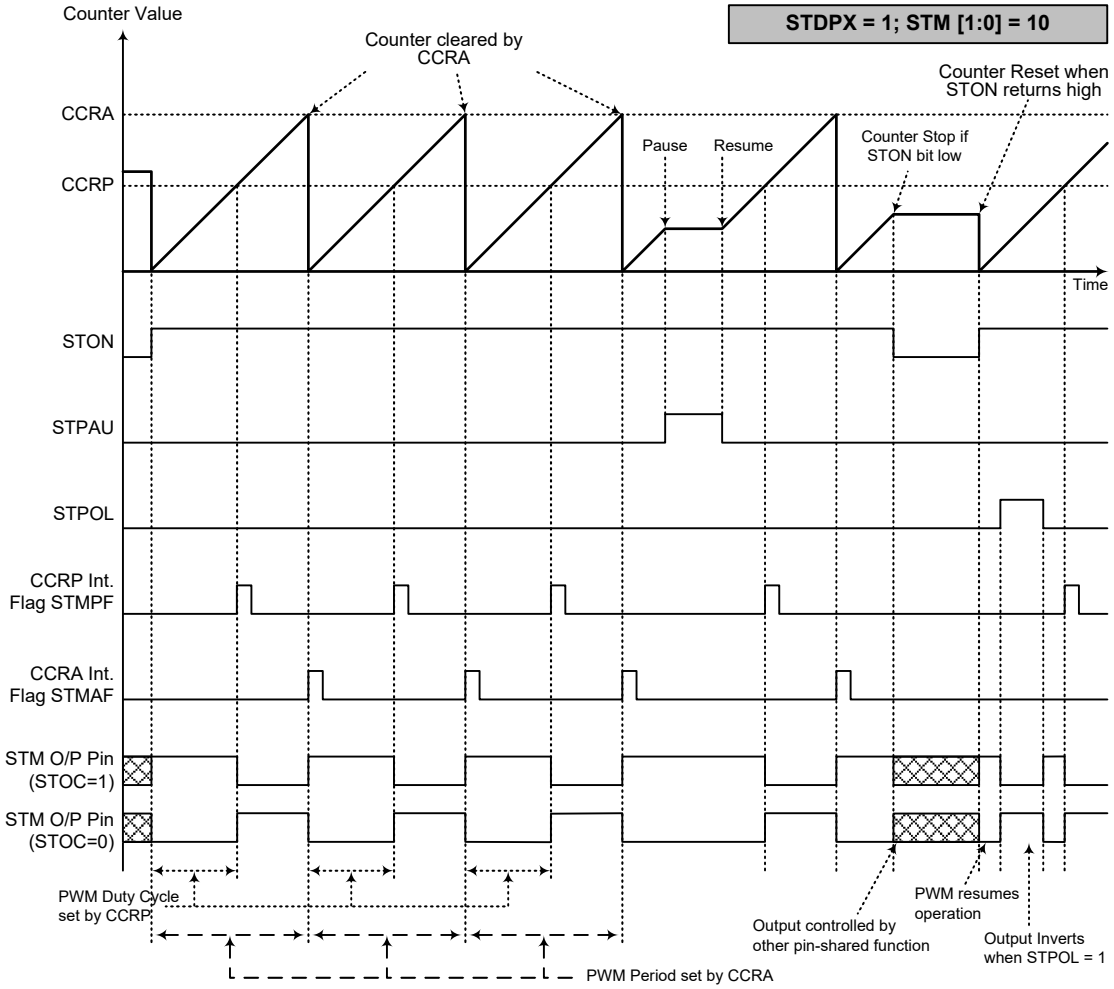
CCRP	1~255	0
Period	CCRA	
Duty	CCRP×256	65536

The PWM output period is determined by the CCRA register value together with the STM clock while the PWM duty cycle is defined by the CCRP register value except when the CCRP value is equal to 0.



PWM Output Mode – STDPX=0

Note: 1. Here STDPX=0 – Counter cleared by CCRP
2. A counter clear sets the PWM Period
3. The internal PWM function continues running even when STIO[1:0]=00 or 01
4. The STCCLR bit has no influence on PWM operation



PWM Output Mode – STDPX=1

Note: 1. Here STDPX=1 – Counter cleared by CCRA

2. A counter clear sets the PWM Period

3. The internal PWM function continues even when STIO[1:0]=00 or 01

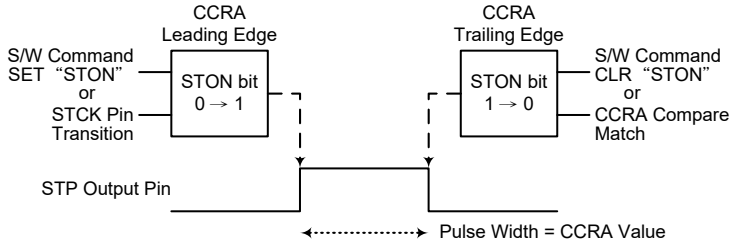
4. The STCCLR bit has no influence on PWM operation

Single Pulse Output Mode

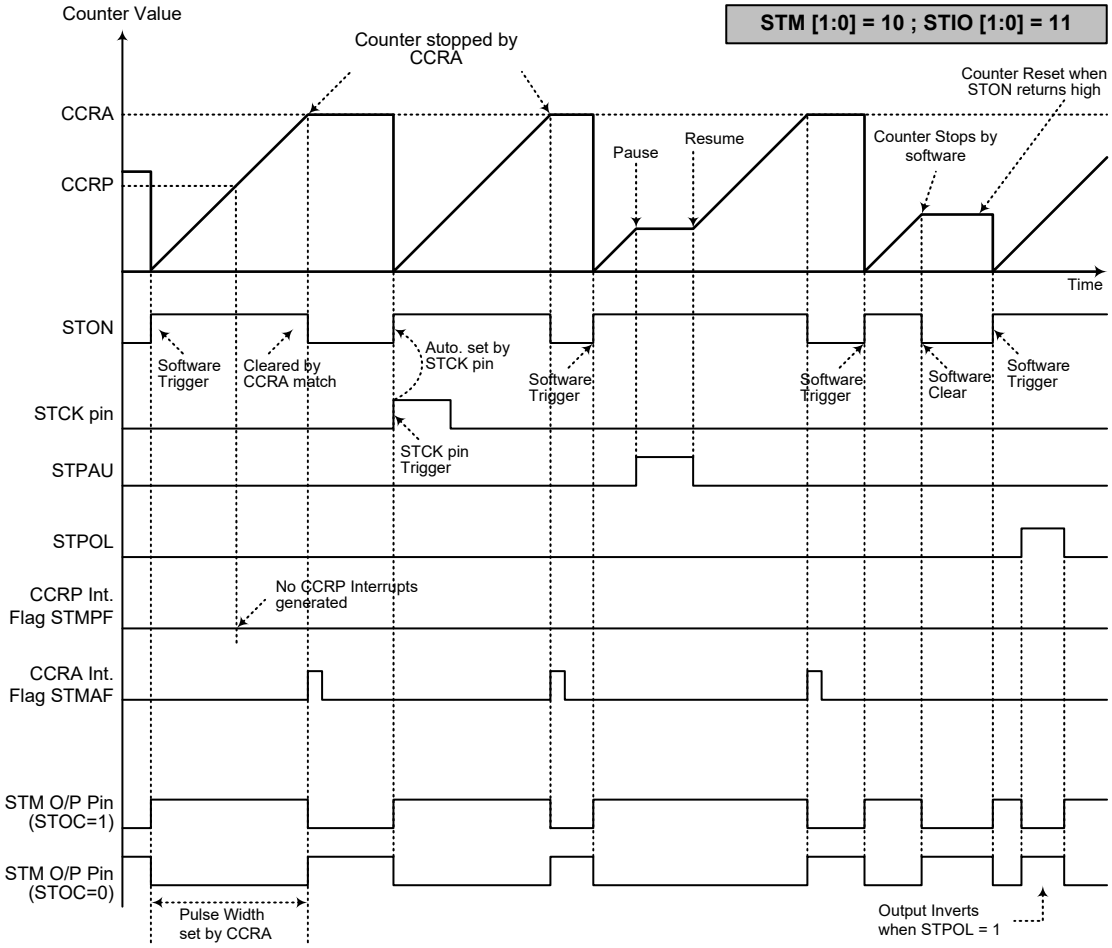
To select this mode, bits STM1 and STM0 in the STMC1 register should be set to 10 respectively and also the STIO1 and STIO0 bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the STM output pin.

The trigger for the pulse output leading edge is a low to high transition of the STON bit, which can be implemented using the application program. However in the Single Pulse Output Mode, the STON bit can also be made to automatically change from low to high using the external STCK pin, which will in turn initiate the Single Pulse output. When the STON bit transitions to a high level, the counter will start running and the pulse leading edge will be generated. The STON bit should remain high when the pulse is in its active state. The generated pulse trailing edge will be generated when the STON bit is cleared to zero, which can be implemented using the application program or when a compare match occurs from Comparator A.

However a compare match from Comparator A will also automatically clear the STON bit and thus generate the Single Pulse output trailing edge. In this way the CCRA value can be used to control the pulse width. A compare match from Comparator A will also generate a STM interrupt. The counter can only be reset back to zero when the STON bit changes from low to high when the counter restarts. In the Single Pulse Output Mode CCRP is not used. The STCCLR and STDPX bits are not used in this Mode.



Single Pulse Generation



Single Pulse Output Mode

Note: 1. Counter stopped by CCRA

2. CCRP is not used

3. The pulse triggered by the STCK pin or by setting the STON bit high

4. A STCK pin active edge will automatically set the STON bit high

5. In the Single Pulse Output Mode, STIO[1:0] must be set to “11” and cannot be changed

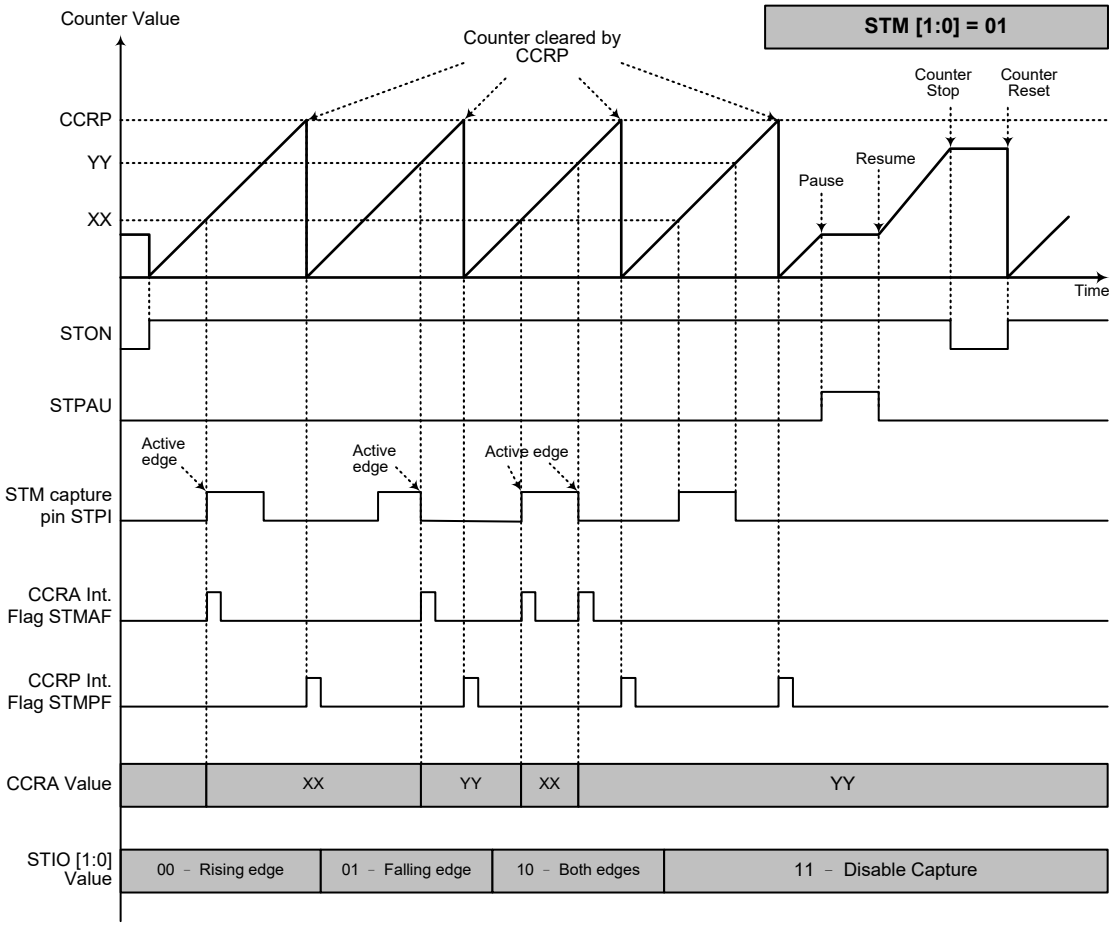
Capture Input Mode

To select this mode bits STM1 and STM0 in the STMC1 register should be set to 01 respectively. This mode enables external signals to capture and store the present value of the internal counter and can therefore be used for applications such as pulse width measurements. The external signal is supplied on the STPI pin, whose active edge can be a rising edge, a falling edge or both rising and falling edges; the active edge transition type is selected using the STIO1 and STIO0 bits in the STMC1 register. The counter is started when the STON bit changes from low to high which is initiated using the application program.

When the required edge transition appears on the STPI pin the present value in the counter will be latched into the CCRA registers and a STM interrupt generated. Irrespective of what events occur on the STPI pin the counter will continue to free run until the STON bit changes from high to low. When a CCRP compare match occurs the counter will reset back to zero; in this way the CCRP value can be used to control the maximum counter value. When a CCRP compare match occurs from Comparator P, a STM interrupt will also be generated. Counting the number of overflow interrupt signals from the CCRP can be a useful method in measuring long pulse widths. The STIO1 and STIO0 bits can select the active trigger edge on the STPI pin to be a rising edge, falling edge or both edge types. If the STIO1 and STIO0 bits are both set high, then no capture operation will take place irrespective of what happens on the STPI pin, however it must be noted that the counter will continue to run.

There are some considerations that should be noted. If the captured pulse width is less than 2 timer clock periods, it may be ignored by hardware. After the counter value is latched to the CCRA registers by an active capture edge, the STMAF flag will be set high after 0.5 timer clock periods. The delay time from the active capture edge received to the action of latching counter value to CCRA registers is less than 1.5 timer clock periods.

The STCCLR and STDPX bits are not used in this Mode.

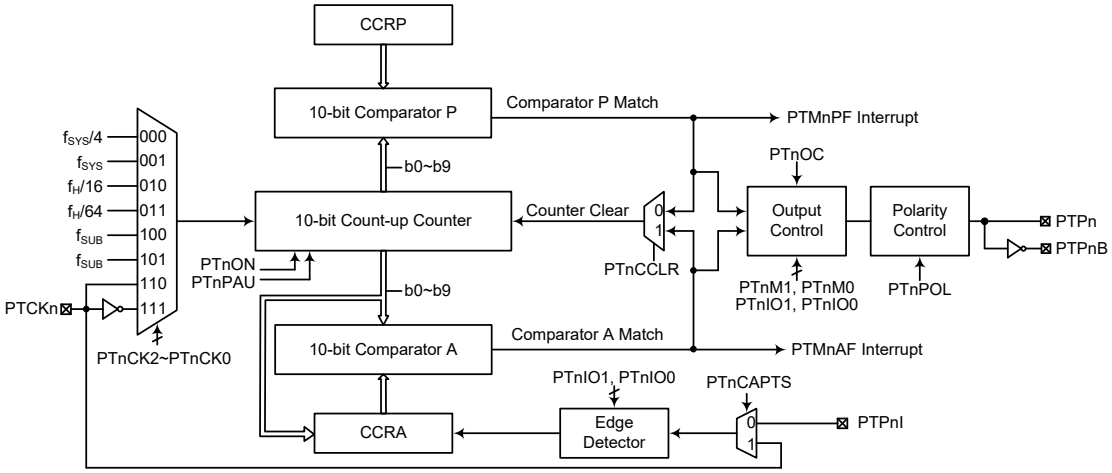


Capture Input Mode

Note: 1. STM[1:0]=01 and active edge set by the STIO[1:0] bits
2. A STM Capture input pin active edge transfers the counter value to CCRA
3. STCCLR bit not used
4. No output function – STOC and STPOL bits are not used
5. CCRP determines the counter value and the counter has a maximum count value when CCRP is equal to zero
6. The capture input mode cannot be used if the selected STM counter clock is not available

Periodic Type TM – PTM

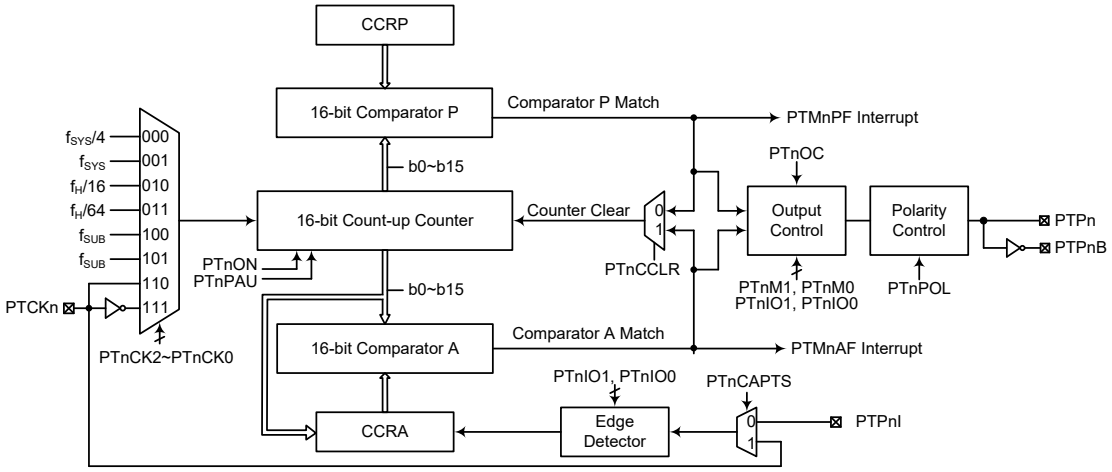
The Periodic Type TM contains five operating modes, which are Compare Match Output, Timer/Event Counter, Capture Input, Single Pulse Output and PWM Output modes. The Periodic TM can also be controlled with up to two external input pins and can drive up to two external output pins.



Note: 1. The PTMn external pins are pin-shared with other functions, so before using the PTMn function the pin-shared function registers must be set properly to enable the PTMn pin function. The PTCKn and PTPnI pins, if used, must also be set as an input by setting the corresponding bits in the port control register.

2. PTP0I, PTP0B, PTCK1 and PTP1I are not externally bounded.

10-bit Periodic Type TM Block Diagram (n=0~1)



Note: The PTMn external pins are pin-shared with other functions, so before using the PTMn function the pin-shared function registers must be set properly to enable the PTMn pin function. The PTCKn and PTPnI pins, if used, must also be set as an input by setting the corresponding bits in the port control register.

16-bit Periodic Type TM Block Diagram (n=2~3)

Periodic TM Operation

The size of Periodic Type TM is 10-/16-bit wide and its core is a 10-/16-bit count-up counter which is driven by a user selectable internal or external clock source. There are also two internal comparators with the names, Comparator A and Comparator P. These comparators will compare the value in the counter with CCRP and CCRA registers. The CCRP and CCRA comparators are 10-/16-bit wide whose value is respectively compared with all counter bits.

The only way of changing the value of the 10-/16-bit counter using the application program is to clear the counter by changing the PTnON bit from low to high. The counter will also be cleared automatically by a counter overflow or a compare match with one of its associated comparators. When these conditions occur, a PTMn interrupt signal will also usually be generated. The Periodic Type TM can operate in a number of different operational modes, can be driven by different clock sources including an input pin and can also control up to two output pins. All operating setup conditions are selected using relevant internal registers.

Periodic Type TM Register Description

Overall operation of the Periodic TM is controlled using a series of registers. A read only register pair exists to store the internal counter 10-/16-bit value, while two read/write register pairs exist to store the internal 10-/16-bit CCRA and CCRP value. The remaining two registers are control registers which setup the different operating and control modes.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PTMnC0	PTnPAU	PTnCK2	PTnCK1	PTnCK0	PTnON	—	—	—
PTMnC1	PTnM1	PTnM0	PTnIO1	PTnIO0	PTnOC	PTnPOL	PTnCAPTS	PTnCCLR
PTMnDL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnDH	—	—	—	—	—	—	D9	D8
PTMnAL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnAH	—	—	—	—	—	—	D9	D8
PTMnRPL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnRPH	—	—	—	—	—	—	D9	D8

10-bit Periodic TM Register List (n=0~1)

Register Name	Bit							
	7	6	5	4	3	2	1	0
PTMnC0	PTnPAU	PTnCK2	PTnCK1	PTnCK0	PTnON	—	—	—
PTMnC1	PTnM1	PTnM0	PTnIO1	PTnIO0	PTnOC	PTnPOL	PTnCAPTS	PTnCCLR
PTMnDL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnDH	D15	D14	D13	D12	D11	D10	D9	D8
PTMnAL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnAH	D15	D14	D13	D12	D11	D10	D9	D8
PTMnRPL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnRPH	D15	D14	D13	D12	D11	D10	D9	D8

16-bit Periodic TM Register List (n=2~3)

- **PTMnC0 Register (n=0~3)**

Bit	7	6	5	4	3	2	1	0
Name	PTnPAU	PTnCK2	PTnCK1	PTnCK0	PTnON	—	—	—
R/W	R/W	R/W	R/W	R/W	R/W	—	—	—
POR	0	0	0	0	0	—	—	—

Bit 7 **PTnPAU:** PTMn Counter Pause control
0: Run
1: Pause

The counter can be paused by setting this bit high. Clearing the bit to zero restores normal counter operation. When in a Pause condition the PTMn will remain powered up and continue to consume power. The counter will retain its residual value when this bit changes from low to high and resume counting from this value when the bit changes to a low value again.

Bit 6~4 **PTnCK2~PTnCK0:** Select PTMn Counter clock
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{SUB}
101: f_{SUB}
110: PTCKn rising edge clock – Unavailable for the PTM1
111: PTCKn falling edge clock – Unavailable for the PTM1

These three bits are used to select the clock source for the PTMn. The external pin clock source can be chosen to be active on the rising or falling edge. The clock source f_{SYS} is the system clock, while f_H and f_{SUB} are other internal clocks, the details of which can be found in the oscillator section.

Bit 3 **PTnON:** PTMn Counter On/Off control
0: Off
1: On

This bit controls the overall on/off function of the PTMn. Setting the bit high enables the counter to run while clearing the bit disables the PTMn. Clearing this bit to zero will stop the counter from counting and turn off the PTMn which will reduce its power consumption. When the bit changes state from low to high the internal counter value will be reset to zero, however when the bit changes from high to low, the internal counter will retain its residual value until the bit returns high again.

If the PTMn is in the Compare Match Output Mode, PWM Output Mode or Single Pulse Output Mode then the PTMn output pin will be reset to its initial condition, as specified by the PTnOC bit, when the PTnON bit changes from low to high.

Bit 2~0 Unimplemented, read as "0"

• PTMnC1 Register (n=0~3)

Bit	7	6	5	4	3	2	1	0
Name	PTnM1	PTnM0	PTnIO1	PTnIO0	PTnOC	PTnPOL	PTnCAPTS	PTnCCLR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PTnM1~PTnM0:** Select PTMn Operating Mode
00: Compare Match Output Mode
01: Capture Input Mode – Unavailable for the PTM1
10: PWM Output Mode or Single Pulse Output Mode
11: Timer/Counter Mode

These bits setup the required operating mode for the PTMn. To ensure reliable operation the PTMn should be switched off before any changes are made to the PTnM1 and PTnM0 bits. In the Timer/Counter Mode, the PTMn output pin state is undefined.

Bit 5~4 **PTnIO1~PTnIO0:** Select PTMn external pin function

Compare Match Output Mode
00: No change
01: Output low
10: Output high
11: Toggle output

PWM Output Mode/Single Pulse Output Mode
00: PWM output inactive state
01: PWM output active state
10: PWM output
11: Single Pulse Output

Capture Input Mode – Unavailable for the PTM1
00: Input capture at rising edge of PTPnI or PTCKn
01: Input capture at falling edge of PTPnI or PTCKn
10: Input capture at rising/falling edge of PTPnI or PTCKn
11: Input capture disabled

Timer/Counter Mode
Unused

These two bits are used to determine how the PTMn external pin changes state when a certain condition is reached. The function that these bits select depends upon in which mode the PTMn is running.

In the Compare Match Output Mode, the PTnIO1 and PTnIO0 bits determine how the PTMn output pin changes state when a compare match occurs from the Comparator A. The PTMn output pin can be setup to switch high, switch low or to toggle its present state when a compare match occurs from the Comparator A. When the bits are both zero, then no change will take place on the output. The initial value of the PTMn output pin should be setup using the PTnOC bit in the PTMnC1 register. Note that the output level requested by the PTnIO1 and PTnIO0 bits must be different from the initial value setup using the PTnOC bit otherwise no change will occur on the PTMn output pin when a compare match occurs. After the PTMn output pin changes state, it can be reset to its initial level by changing the level of the PTnON bit from low to high.

In the PWM Output Mode, the PTnIO1 and PTnIO0 bits determine how the TM output pin changes state when a certain compare match condition occurs. The PTMn output function is modified by changing these two bits. It is necessary to only change the values of the PTnIO1 and PTnIO0 bits only after the PTMn has been switched off. Unpredictable PWM outputs will occur if the PTnIO1 and PTnIO0 bits are changed when the PTMn is running.

Bit 3 **PTnOC:** PTMn PTPn Output control

Compare Match Output Mode
0: Initial low
1: Initial high

PWM Output Mode/Single Pulse Output Mode
0: Active low
1: Active high

This is the output control bit for the PTMn output pin. Its operation depends upon whether PTMn is being used in the Compare Match Output Mode or in the PWM Output Mode/Single Pulse Output Mode. It has no effect if the PTMn is in the Timer/Counter Mode. In the Compare Match Output Mode it determines the logic level of the PTMn output pin before a compare match occurs. In the PWM Output Mode it determines if the PWM signal is active high or active low. In the Single Pulse Output Mode it determines the logic level of the PTMn output pin when the PTnON bit changes from low to high.

Bit 2 **PTnPOL:** PTMn PTPn Output polarity control
0: Non-inverted
1: Inverted

This bit controls the polarity of the PTPn output pin. When the bit is set high the PTMn output pin will be inverted and not inverted when the bit is zero. It has no effect if the PTMn is in the Timer/Counter Mode.

Bit 1 **PTnCAPTS:** PTMn Capture Trigger Source selection
0: From PTPnI pin
1: From PTCKn pin
Note that for the PTM0, this bit must be set to "1" when the capture input mode is used. For the PTM1, this bit should be kept unchanged after power on.

Bit 0 **PTnCCLR:** PTMn Counter Clear condition selection
0: Comparator P match
1: Comparator A match
This bit is used to select the method which clears the counter. Remember that the Periodic TM contains two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the PTnCCLR bit set high, the counter will be cleared when a compare match occurs from the Comparator A. When the bit is low, the counter will be cleared when a compare match occurs from the Comparator P or with a counter overflow. A counter overflow clearing method can only be implemented if the CCRP bits are all cleared to zero. The PTnCCLR bit is not used in the PWM Output, Single Pulse Output or Capture Input Mode.

• PTMnDL Register (n=0~3)

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0:** PTMn Counter Low Byte Register bit 7 ~ bit 0
PTMn 10-/16-bit Counter bit 7 ~ bit 0

• PTMnDH Register (n=0~1)

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	D9	D8
R/W	—	—	—	—	—	—	R	R
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"
Bit 1~0 **D9~D8:** PTMn Counter High Byte Register bit 1 ~ bit 0
PTMn 10-bit Counter bit 9 ~ bit 8

• PTMnDH Register (n=2~3)

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D15~D8:** PTMn Counter High Byte Register bit 7 ~ bit 0
PTMn 16-bit Counter bit 15 ~ bit 8

• PTMnAL Register (n=0~3)

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0:** PTMn CCRA Low Byte Register bit 7 ~ bit 0
PTMn 10-/16-bit CCRA bit 7 ~ bit 0

• **PTMnAH Register (n=0~1)**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	D9	D8
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **D9~D8:** PTMn CCRA High Byte Register bit 1 ~ bit 0
PTMn 10-bit CCRA bit 9 ~ bit 8

• **PTMnAH Register (n=2~3)**

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D15~D8:** PTMn CCRA High Byte Register bit 7 ~ bit 0
PTMn 16-bit CCRA bit 15 ~ bit 8

• **PTMnRPL Register (n=0~3)**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0:** PTMn CCRP Low Byte Register bit 7 ~ bit 0
PTMn 10-/16-bit CCRP bit 7 ~ bit 0

• **PTMnRPH Register (n=0~1)**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	D9	D8
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **D9~D8:** PTMn CCRP High Byte Register bit 1 ~ bit 0
PTMn 10-bit CCRP bit 9 ~ bit 8

• **PTMnRPH Register (n=2~3)**

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D15~D8:** PTMn CCRP High Byte Register bit 7 ~ bit 0
PTMn 16-bit CCRP bit 15 ~ bit 8

Periodic Type TM Operation Modes

The Periodic Type TM can operate in one of five operating modes, Compare Match Output Mode, PWM Output Mode, Single Pulse Output Mode, Capture Input Mode or Timer/Counter Mode. The operating mode is selected using the PTnM1 and PTnM0 bits in the PTMnC1 register.

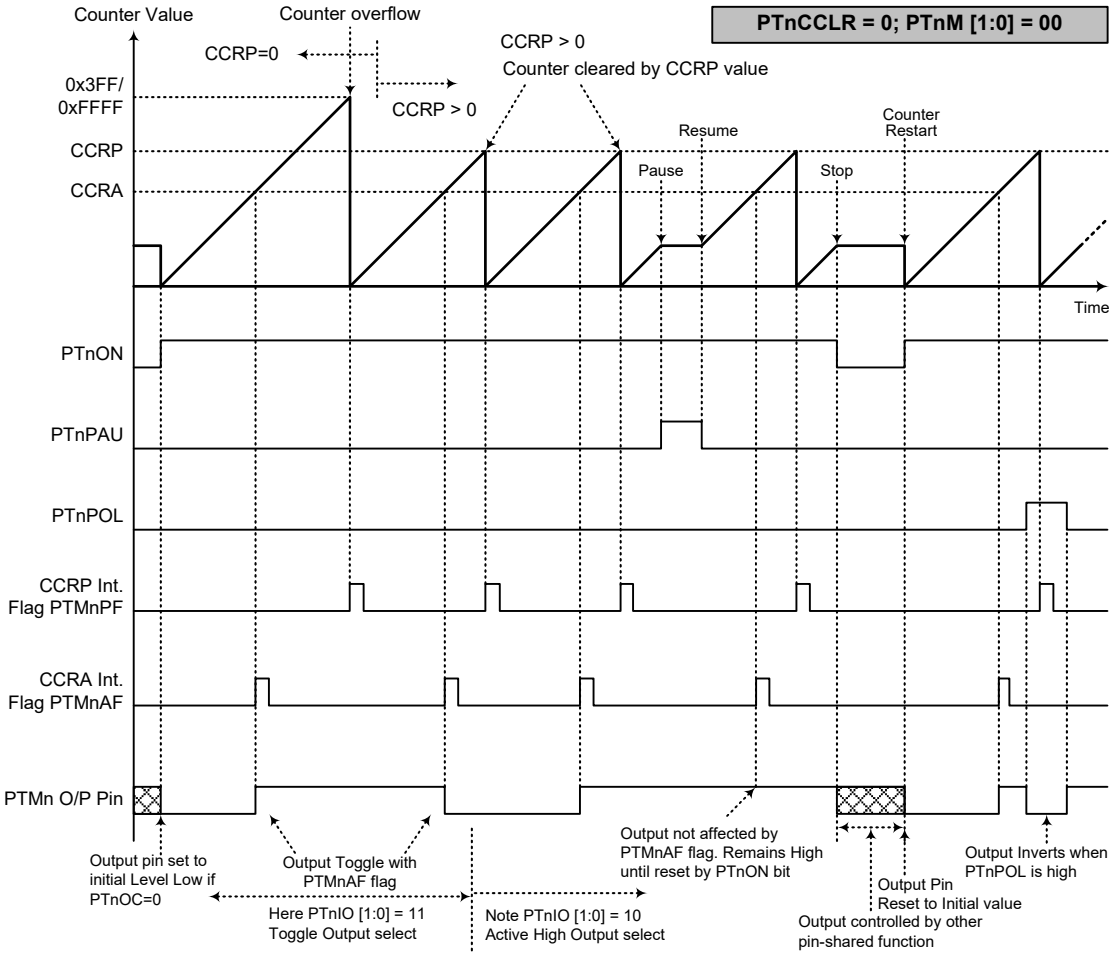
Compare Match Output Mode

To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 register, should be set to 00 respectively. In this mode once the counter is enabled and running it can be cleared by three methods. These are a counter overflow, a compare match from Comparator A and a compare match from Comparator P. When the PTnCCLR bit is low, there are two ways in which the counter can be cleared. One is when a compare match from Comparator P, the other is when the CCRP bits are all zero which allows the counter to overflow. Here both PTMnAF and PTMnPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

If the PTnCCLR bit in the PTMnC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the PTMnAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when PTnCCLR is high no PTMnPF interrupt request flag will be generated. In the Compare Match Output Mode, the CCRA cannot be cleared to “0”.

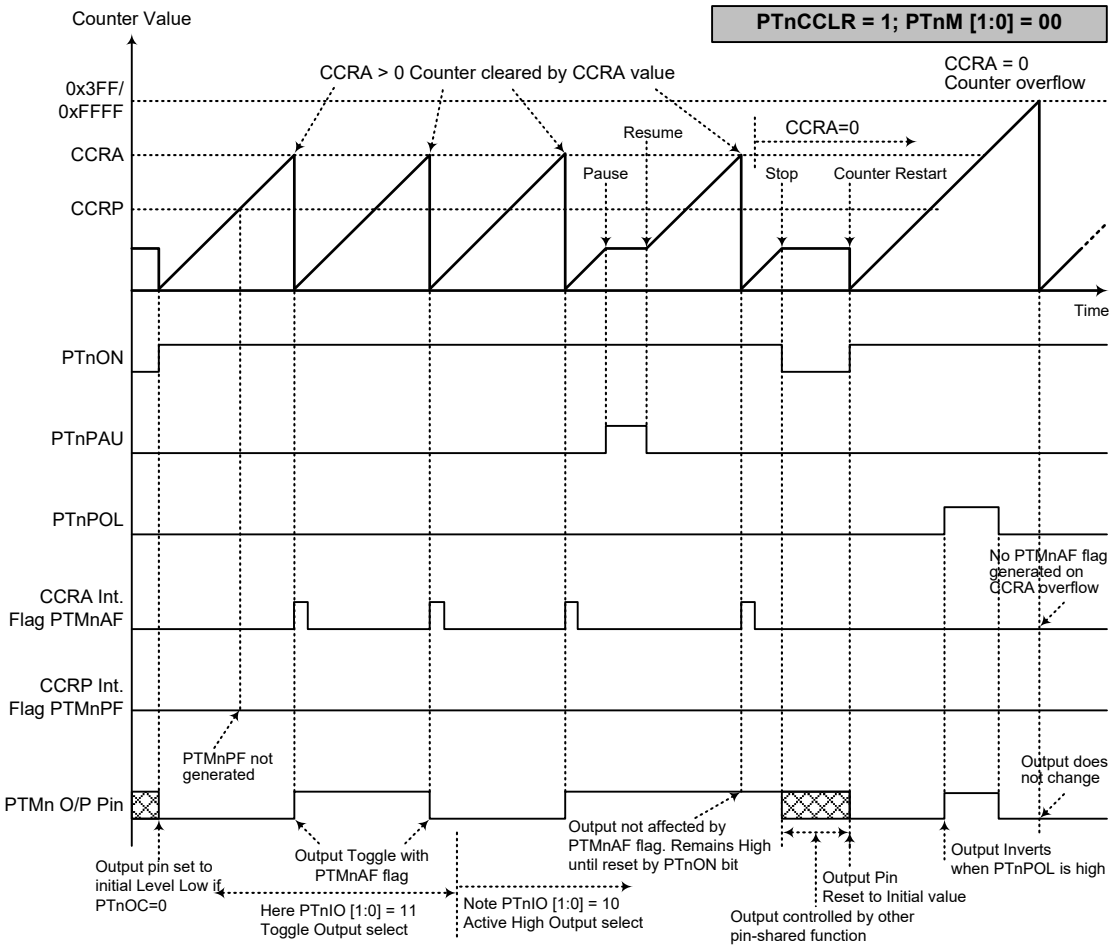
If the CCRA bits are all zero, the counter will overflow when it reaches its maximum 10-bit, 3FF Hex, or 16-bit, FFFF Hex, value, however here the PTMnAF interrupt request flag will not be generated.

As the name of the mode suggests, after a comparison is made, the PTMn output pin will change state. The PTMn output pin condition however only changes state when a PTMnAF interrupt request flag is generated after a compare match occurs from Comparator A. The PTMnPF interrupt request flag, generated from a compare match occurs from Comparator P, will have no effect on the PTMn output pin. The way in which the PTMn output pin changes state are determined by the condition of the PTnIO1 and PTnIO0 bits in the PTMnC1 register. The PTMn output pin can be selected using the PTnIO1 and PTnIO0 bits to go high, to go low or to toggle from its present condition when a compare match occurs from Comparator A. The initial condition of the PTMn output pin, which is setup after the PTnON bit changes from low to high, is setup using the PTnOC bit. Note that if the PTnIO1 and PTnIO0 bits are zero then no pin change will take place.



Compare Match Output Mode – PTnCCLR=0

Note: 1. With PTnCCLR=0, a Comparator P match will clear the counter
2. The PTMn output pin is controlled only by the PTMnAF flag
3. The output pin is reset to its initial state by a PTnON bit rising edge
4. The 10-bit PTM maximum counter value is 0x3FF while the 16-bit PTM maximum counter value is 0xFFFF
5. n=0~1 for 10-bit PTM while n=2~3 for 16-bit PTM



Compare Match Output Mode – PTnCCLR=1

Note: 1. With PTnCCLR=1, a Comparator A match will clear the counter
2. The PTMn output pin is controlled only by the PTMnAF flag
3. The output pin is reset to its initial state by a PTnON bit rising edge
4. A PTMnPF flag is not generated when PTnCCLR=1
5. The 10-bit PTM maximum counter value is 0x3FF while the 16-bit PTM maximum counter value is 0xFFFF
6. n=0~1 for 10-bit PTM while n=2~3 for 16-bit PTM

Timer/Counter Mode

To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 register should be set to 11 respectively. The Timer/Counter Mode operates in an identical way to the Compare Match Output Mode generating the same interrupt flags. The exception is that in the Timer/Counter Mode the PTMn output pin is not used. Therefore the above description and Timing Diagrams for the Compare Match Output Mode can be used to understand its function. As the PTMn output pin is not used in this mode, the pin can be used as a normal I/O pin or other pin-shared function.

PWM Output Mode

To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 register should be set to 10 respectively and also the PTnIO1 and PTnIO0 bits should be set to 10 respectively. The PWM function within the PTMn is useful for applications which require functions such as motor control, heating control, illumination control, etc. By providing a signal of fixed frequency but of varying duty cycle on the PTMn output pin, a square wave AC waveform can be generated with varying equivalent DC RMS values.

As both the period and duty cycle of the PWM waveform can be controlled, the choice of generated waveform is extremely flexible. In the PWM Output Mode, the PTnCCLR bit has no effect as the PWM period. Both of the CCRP and CCRA registers are used to generate the PWM waveform, the CCRP register is used to clear the internal counter and thus control the PWM waveform frequency, while the CCRA register is used to control the duty cycle. The PWM waveform frequency and duty cycle can therefore be controlled by the values in the CCRA and CCRP registers.

An interrupt flag, one for each of the CCRA and CCRP, will be generated when a compare match occurs from either Comparator A or Comparator P. The PTnOC bit in the PTMnC1 register is used to select the required polarity of the PWM waveform while the two PTnIO1 and PTnIO0 bits are used to enable the PWM output or to force the PTMn output pin to a fixed high or low level. The PTnPOL bit is used to reverse the polarity of the PWM output waveform.

• 10-bit PTMn, PWM Output Mode, Edge-aligned Mode (n=0~1)

CCRP	1~1023	0
Period	1~1023	1024
Duty	CCRA	

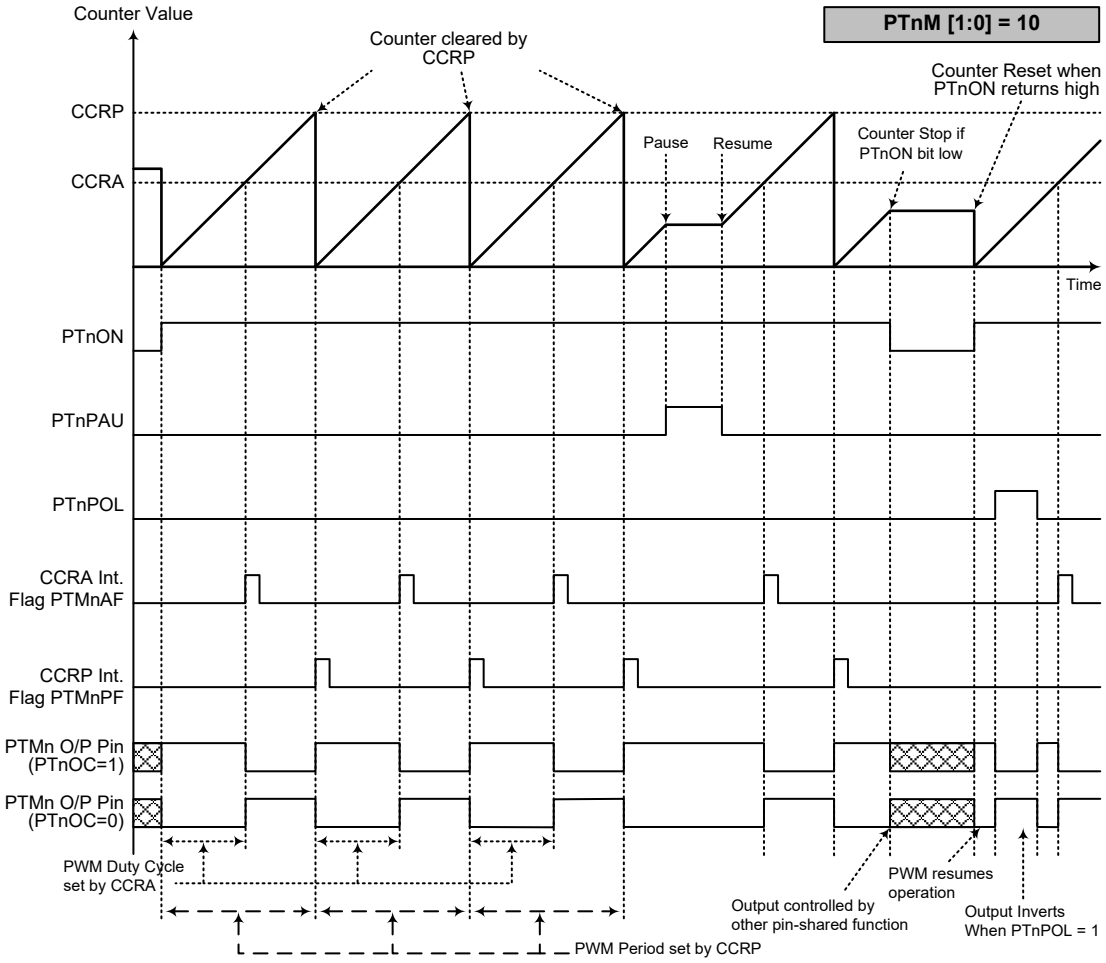
• 16-bit PTMn, PWM Output Mode, Edge-aligned Mode (n=2~3)

CCRP	1~65535	0
Period	1~65535	65536
Duty	CCRA	

If f_{SYS}=16MHz, PTMn clock source select f_{SYS}/4, CCRP=512 and CCRA=128,

The PTMn PWM output frequency = (f_{SYS}/4)/512=f_{SYS}/2048=8kHz, duty=128/512=25%,

If the Duty value defined by the CCRA register is equal to or greater than the Period value, then the PWM output duty is 100%.



PWM Output Mode

Note: 1. The counter is cleared by CCRP

2. A counter clear sets the PWM Period

3. The internal PWM function continues running even when PTnIO[1:0]=00 or 01

4. The PTnCCLR bit has no influence on PWM operation

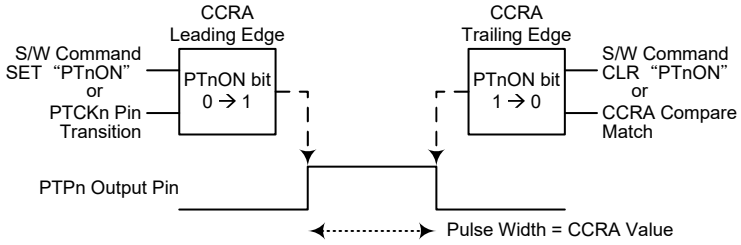
5. n=0~1 for 10-bit PTM while n=2~3 for 16-bit PTM

Single Pulse Output Mode

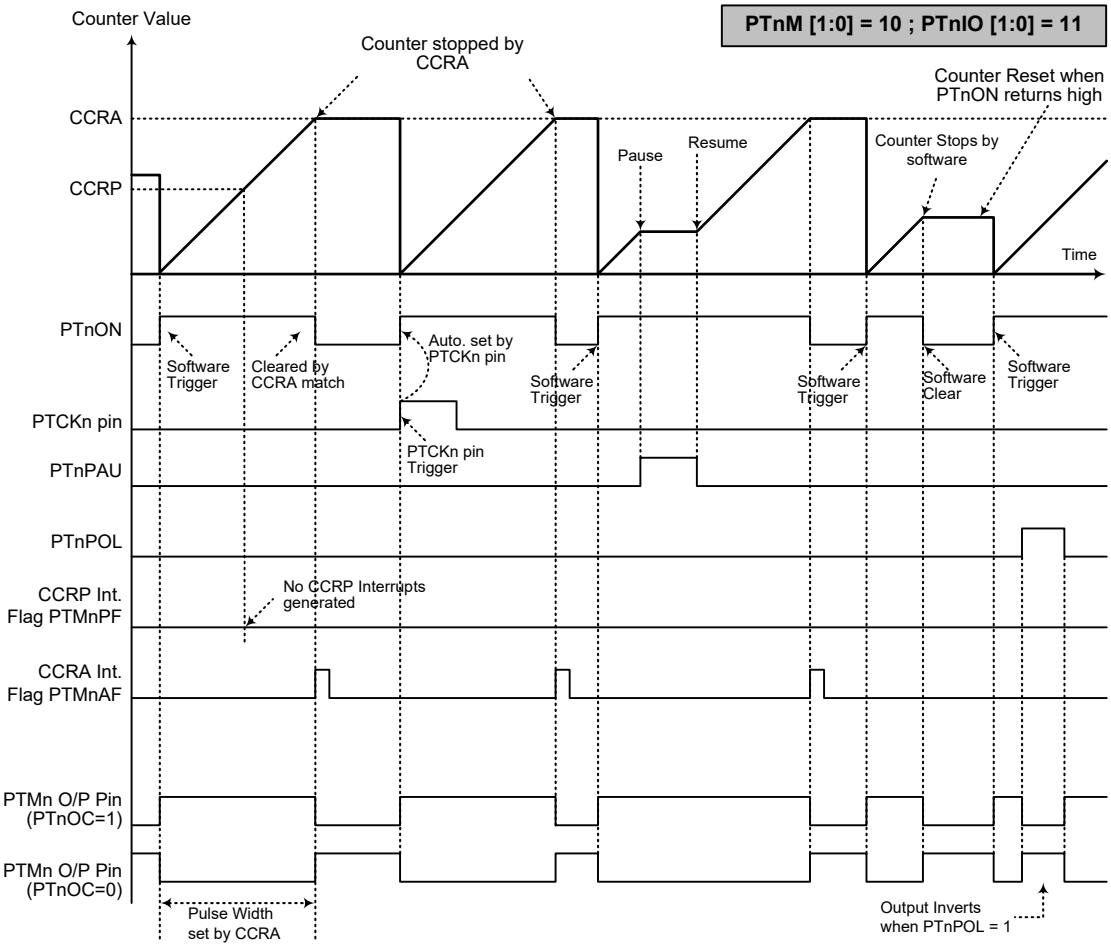
To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 register should be set to 10 respectively and also the PTnIO1 and PTnIO0 bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the PTMn output pin.

The trigger for the pulse output leading edge is a low to high transition of the PTnON bit, which can be implemented using the application program. However in the Single Pulse Output Mode, the PTnON bit can also be made to automatically change from low to high using the external PTCKn pin, which will in turn initiate the Single Pulse output. When the PTnON bit transitions to a high level, the counter will start running and the pulse leading edge will be generated. The PTnON bit should remain high when the pulse is in its active state. The generated pulse trailing edge will be generated when the PTnON bit is cleared to zero, which can be implemented using the application program or when a compare match occurs from Comparator A.

However a compare match from Comparator A will also automatically clear the PTnON bit and thus generate the Single Pulse output trailing edge. In this way the CCRA value can be used to control the pulse width. A compare match from Comparator A will also generate a PTMn interrupt. The counter can only be reset back to zero when the PTnON bit changes from low to high when the counter restarts. In the Single Pulse Output Mode CCRP is not used. The PTnCCLR is not used in this Mode.



Single Pulse Generation



Single Pulse Output Mode

Note: 1. Counter stopped by CCRA

2. CCRP is not used

3. The pulse triggered by the PTCKn pin or by setting the PTnON bit high

4. A PTCKn pin active edge will automatically set the PTnON bit high

5. In the Single Pulse Output Mode, PTnIO[1:0] must be set to “11” and cannot be changed

6. n=0~1 for 10-bit PTM while n=2~3 for 16-bit PTM

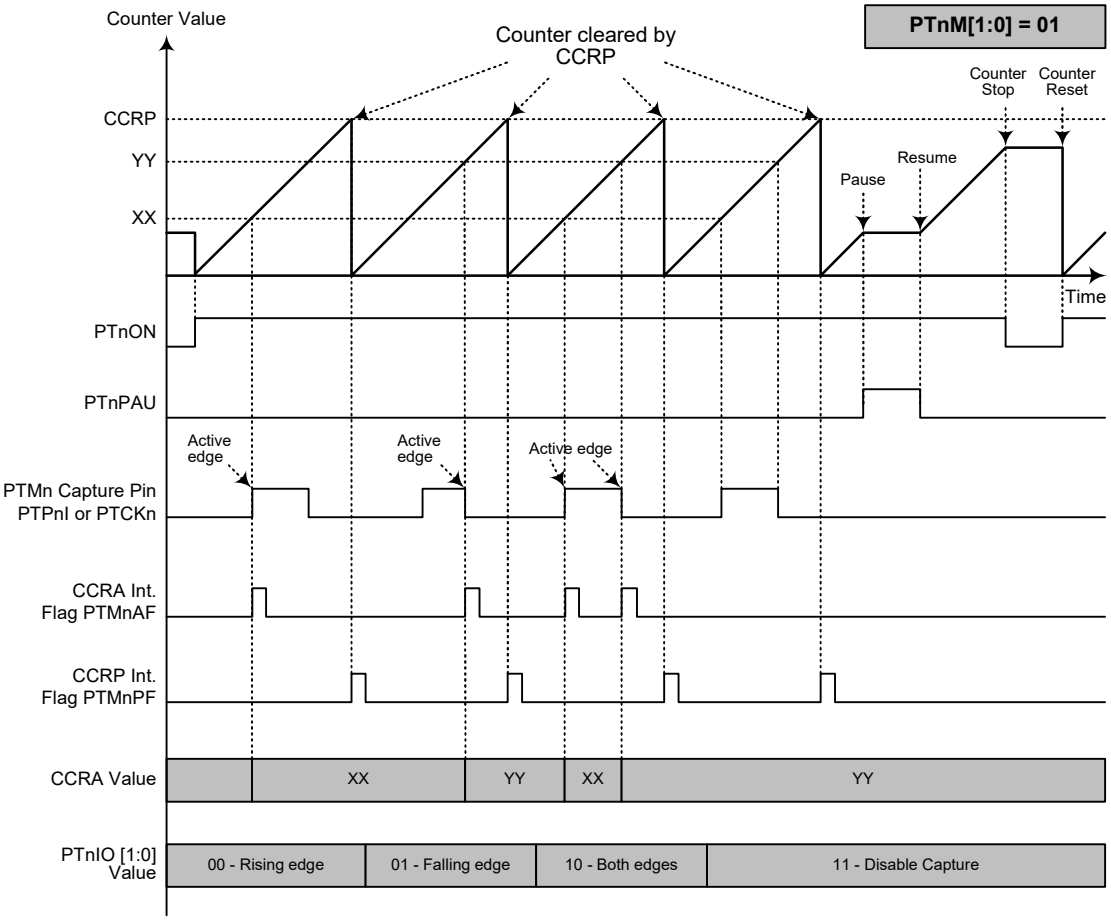
Capture Input Mode

To select this mode bits PTnM1 and PTnM0 in the PTMnC1 register should be set to 01 respectively. This mode enables external signals to capture and store the present value of the internal counter and can therefore be used for applications such as pulse width measurements. The external signal is supplied on the PTPnI or PTCKn pin, selected by the PTnCAPTS bit in the PTMnC1 register. The input pin active edge can be either a rising edge, a falling edge or both rising and falling edges; the active edge transition type is selected using the PTnIO1 and PTnIO0 bits in the PTMnC1 register. The counter is started when the PTnON bit changes from low to high which is initiated using the application program.

When the required edge transition appears on the PTPnI or PTCKn pin the present value in the counter will be latched into the CCRA registers and a PTMn interrupt generated. Irrespective of what events occur on the PTPnI or PTCKn pin the counter will continue to free run until the PTnON bit changes from high to low. When a CCRP compare match occurs the counter will reset back to zero; in this way the CCRP value can be used to control the maximum counter value. When a CCRP compare match occurs from Comparator P, a PTMn interrupt will also be generated. Counting the number of overflow interrupt signals from the CCRP can be a useful method in measuring long pulse widths. The PTnIO1 and PTnIO0 bits can select the active trigger edge on the PTPnI or PTCKn pin to be a rising edge, falling edge or both edge types. If the PTnIO1 and PTnIO0 bits are both set high, then no capture operation will take place irrespective of what happens on the PTPnI or PTCKn pin, however it must be noted that the counter will continue to run.

There are some considerations that should be noted. If PTCKn is used as the capture input source, then it cannot be selected as the PTMn clock source. If the captured pulse width is less than 2 timer clock periods, it may be ignored by hardware. After the counter value is latched to the CCRA registers by an active capture edge, the PTMnAF flag will be set high after 0.5 timer clock periods. The delay time from the active capture edge received to the action of latching counter value to CCRA registers is less than 1.5 timer clock periods.

The PTnCCLR, PTnOC and PTnPOL bits are not used in this Mode.



Capture Input Mode

Note: 1. PTnM[1:0]=01 and active edge set by the PTnIO[1:0] bits
2. A PTMn Capture input pin active edge transfers the counter value to CCRA
3. PTnCCLR bit not used
4. No output function – PTnOC and PTnPOL bits are not used
5. CCRP determines the counter value and the counter has a maximum count value when CCRP is equal to zero
6. The capture input mode cannot be used if the selected PTMn counter clock is not available
7. n=0~1 for 10-bit PTM while n=2~3 for 16-bit PTM

Analog to Digital Converter

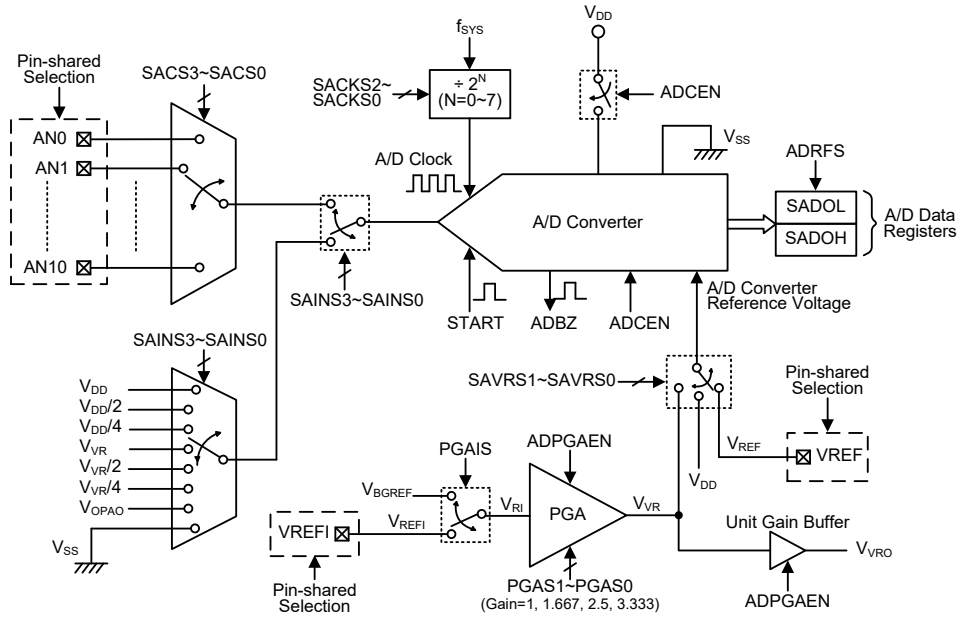
The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a microcontroller, they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the microcontroller, the need for external components is reduced significantly with the corresponding follow-on benefits of lower costs and reduced component space requirements.

A/D Converter Overview

The device contains a multi-channel analog to digital converter which can directly interface to external analog signals, such as that from sensors or other control signals and convert these signals directly into a 12-bit digital value. It also can convert the internal signals, such as the internal reference voltage, into a 12-bit digital value. The external or internal analog signal to be converted is determined by the SAINS3~SAINS0 and SACS3~SACS0 bits. Note that when the internal analog signal is selected to be converted using the SAINS3~SAINS0 bits, the external channel analog input will automatically be switched off. More detailed information about the A/D input signal selection will be described in the "A/D Converter Input Signals" section.

External Input Channels	Internal Signal	A/D Signal Select
AN0~AN10	V_{DD}, $V_{DD}/2$, $V_{DD}/4$, V_{VR}, $V_{VR}/2$, $V_{VR}/4$, V_{OPAO}, V_{SS}	SAINS3~SAINS0 SACS3~SACS0

The accompanying block diagram shows the internal structure of the A/D converter with temperature sensor together with its associated registers and control bits.



A/D Converter Structure

A/D Register Description

Overall operation of the A/D converter is controlled using six registers. A read only register pair exists to store the A/D Converter data 12-bit value. Three registers, SADC0, SADC1 and SADC2, are the control registers which setup the operating conditions and control function of the A/D converter. The VBGRC register contains the VBGREN bit to control the bandgap reference voltage.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SADOL (ADRFS=0)	D3	D2	D1	D0	—	—	—	—
SADOL (ADRFS=1)	D7	D6	D5	D4	D3	D2	D1	D0
SADOH (ADRFS=0)	D11	D10	D9	D8	D7	D6	D5	D4
SADOH (ADRFS=1)	—	—	—	—	D11	D10	D9	D8
SADC0	START	ADBZ	ADCEN	ADRFS	SACS3	SACS2	SACS1	SACS0
SADC1	SAINS3	SAINS2	SAINS1	SAINS0	—	SACKS2	SACKS1	SACKS0
SADC2	ADPGAEN	—	—	PGAIS	SAVRS1	SAVRS0	PGAGS1	PGAGS0
VBGRC	—	—	—	—	—	—	—	VBGREN

A/D Converter Register List

A/D Converter Data Registers – SADOL, SADOH

As the device contains an internal 12-bit A/D converter, it requires two data registers to store the converted value. These are a high byte register, known as SADOH, and a low byte register, known as SADOL. After the conversion process takes place, these registers can be directly read by the microcontroller to obtain the digitised conversion value. As only 12 bits of the 16-bit register space is utilised, the format in which the data is stored is controlled by the ADRFS bit in the SADC0 register as shown in the accompanying table. D0~D11 are the A/D conversion result data bits. Any unused bits will be read as zero. The A/D data registers contents will be unchanged if the A/D converter is disabled.

ADRFS	SADOH								SADOL							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0
1	0	0	0	0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

A/D Converter Data Registers

A/D Converter Control Registers – SADC0, SADC1, SADC2

To control the function and operation of the A/D converter, three control registers known as SADC0, SADC1 and SADC2 are provided. These 8-bit registers define functions such as the selection of which analog signal is connected to the internal A/D converter, the digitised data format, the A/D clock source as well as controlling the start function and monitoring the A/D converter busy status. As the device contains only one actual analog to digital converter hardware circuit, each of the external and internal analog signals must be routed to the converter. The SAINS3~SAINS0 bits in the SADC1 register and SACS3~SACS0 bits in the SADC0 register are used to determine which analog signal derived from the external or internal signals will be connected to the A/D converter. The A/D converter also contains a programmable gain amplifier, PGA, to generate the A/D converter internal reference voltage. The overall operation of the PGA is controlled using the SADC2 register.

The relevant pin-shared function selection bits determine which pins on I/O Ports are used as analog inputs for the A/D converter input and which pins are not. When the pin is selected to be an A/D input, its original function whether it is an I/O or other pin-shared function will be removed. In addition, any internal pull-high resistor connected to the pin will be automatically removed if the pin is selected to be an A/D converter input.

• **SADC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	START	ADBZ	ADCEN	ADRFS	SACS3	SACS2	SACS1	SACS0
R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **START:** Start the A/D Conversion
0→1→0: Start
This bit is used to initiate an A/D conversion process. The bit is normally low but if set high and then cleared low again, the A/D converter will initiate a conversion process.

Bit 6 **ADBZ:** A/D Converter busy flag
0: No A/D conversion is in progress
1: A/D conversion is in progress
This read only flag is used to indicate whether the A/D conversion is in progress or not. When the START bit is set from low to high and then to low again, the ADBZ flag will be set to 1 to indicate that the A/D conversion is initiated. The ADBZ flag will be cleared to 0 after the A/D conversion is complete.

Bit 5 **ADCEN:** A/D Converter function enable control
0: Disable
1: Enable
This bit controls the A/D internal function. This bit should be set to one to enable the A/D converter. If the bit is cleared to zero, then the A/D converter will be switched off reducing the device power consumption. When the A/D converter function is disabled, the contents of the A/D data register pair known as SADOH and SADOL will be unchanged.

Bit 4 **ADRFS:** A/D conversion data format select
0: A/D converter data format → SADOH=D[11:4]; SADOL=D[3:0]
1: A/D converter data format → SADOH=D[11:8]; SADOL=D[7:0]
This bit controls the format of the 12-bit converted A/D value in the two A/D data registers. Details are provided in the A/D converter data register section.

Bit 3~0 **SACS3~SACS0:** A/D converter external analog input channel select
0000: AN0
0001: AN1
0010: AN2
0011: AN3
0100: AN4
0101: AN5
0110: AN6
0111: AN7
1000: AN8
1001: AN9
1010: AN10
1011~1111: Non-existed channel, input floating if selected

• **SADC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	SAINS3	SAINS2	SAINS1	SAINS0	—	SACKS2	SACKS1	SACKS0
R/W	R/W	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	0	0	0	0	—	0	0	0

Bit 7~4 **SAINS3~SAINS0:** A/D converter input signal select
0000: External source – External analog channel input, ANn
0001: Internal source – Internal A/D converter power supply voltage V_{DD}
0010: Internal source – Internal A/D converter power supply voltage $V_{DD}/2$
0011: Internal source – Internal A/D converter power supply voltage $V_{DD}/4$
0100: External source – External analog channel input, ANn

0101: Internal source – Internal A/D converter PGA output voltage V_{VR}
0110: Internal source – Internal A/D converter PGA output voltage $V_{VR}/2$
0111: Internal source – Internal A/D converter PGA output voltage $V_{VR}/4$
1000: Internal source – Internal Operational Amplifier output voltage V_{OPAO}
1001~1011: Internal source – Connected to the ground, V_{SS}
1100~1111: External source – External analog channel input, ANn

When the internal analog signal is selected to be converted, the external channel signal input will automatically be switched off regardless of the SACS3~SACS0 bit values. It will prevent the external channel input from being connected together with the internal analog signal.

Bit 3 Unimplemented, read as "0"

Bit 2~0 **SACKS2~SACKS0:** A/D conversion clock source select
000: f_{SYS}
001: $f_{SYS}/2$
010: $f_{SYS}/4$
011: $f_{SYS}/8$
100: $f_{SYS}/16$
101: $f_{SYS}/32$
110: $f_{SYS}/64$
111: $f_{SYS}/128$

• SADC2 Register

Bit	7	6	5	4	3	2	1	0
Name	ADPGAEN	—	—	PGAIS	SAVRS1	SAVRS0	PGAGS1	PGAGS0
R/W	R/W	—	—	R/W	R/W	R/W	R/W	R/W
POR	0	—	—	0	0	0	0	0

Bit 7 **ADPGAEN:** A/D converter PGA enable/disable control
0: Disable
1: Enable

This bit is used to control the A/D converter internal PGA function. When the PGA output voltage is selected as A/D input or A/D reference voltage, the PGA needs to be enabled by setting this bit high. Otherwise the PGA needs to be disabled by clearing the ADPGAEN bit to zero to conserve power.

Bit 6~5 Unimplemented, read as "0"

Bit 4 **PGAIS:** PGA input voltage selection
0: From VREFI pin
1: From internal reference voltage V_{BGREF}

When the internal independent reference voltage V_{BGREF} is selected as the PGA input, the external reference voltage on the VREFI pin will be automatically switched off. In addition, the internal bandgap reference V_{BGREF} should be enabled by setting the VBGREN bit in the VBGRC register to "1".

Bit 3~2 **SAVRS1~SAVRS0:** A/D converter reference voltage select
00: Internal A/D converter power, V_{DD}
01: External VREF pin
1x: Internal PGA output voltage, V_{VR}

These bits are used to select the A/D converter reference voltage source. When the internal reference voltage source is selected, the reference voltage derived from the external VREF pin will automatically be switched off.

Bit 1~0 **PGAGS1~PGAGS0:** PGA gain select
00: Gain=1
01: Gain=1.667 – V_{VR}=2V as V_{RI}=1.2V
10: Gain=2.5 – V_{VR}=3V as V_{RI}=1.2V
11: Gain=3.333 – V_{VR}=4V as V_{RI}=1.2V

These bits are used to select the PGA gain. Note that here the gain is guaranteed only when the PGA input voltage is equal to 1.2V.

Bandgap Referenc Voltage Control Register – VBGRC

A high performance bandgap voltage reference is included in the device. It has an accurate voltage reference output, V_{BGREF}, when input supply voltage change or temperature variation. The VBGRC register is used to control the bandgap reference voltage circuit enable or disable.

- **VBGRC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	—	VBGREN
R/W	—	—	—	—	—	—	—	R/W
POR	—	—	—	—	—	—	—	0

Bit 7~1 Unimplemented, read as "0"

Bit 0 **VBGREN:** Bandgap reference voltage control
0: Disable
1: Enable
This bit is used to enable the internal Bandgap reference circuit. The internal Bandgap reference circuit should first be enabled before the V_{BGREF} voltage is selected to be used. A specific start-up time is necessary for the Bandgap circuit to become stable and accurate. When this bit is cleared to 0, the Bandgap voltage output V_{BGREF} is in a low state.

A/D Converter Reference Voltage

The actual reference voltage supply to the A/D Converter can be supplied from the internal A/D converter power, V_{DD}, an external reference source supplied on pin VREF or an internal reference source derived from the PGA output V_{VR}. The desired selection is made using the SAVRS1~SAVRS0 bits in the SADC2 register. The internal reference voltage is amplified through a programmable gain amplifier, PGA, which is controlled by the ADPGAEN bit in the SADC2 register. The PGA gain can be equal to 1, 1.667, 2.5 or 3.333 and selected using the PGAGS1~PGAGS0 bits in the SADC2 register. The PGA input can come from the external reference input pin, VREFI, or an internal Bandgap reference voltage, V_{BGREF}, selected by the PGAIS bit in the SADC2 register. The internal Bandgap reference circuit should first be enabled before the V_{BGREF} is selected to be used. A specific start-up time is necessary for the Bandgap circuit to become stable and accurate.

As the VREFI and VREF pins both are pin-shared with other functions, when the VREFI or VREF pin is selected as the reference voltage pin, the VREFI or VREF pin-shared function selection bits should first be properly configured to disable other pin-shared functions. However, if the internal reference signal is selected as the reference source, the external reference input from the VREFI or VREF pin will automatically be switched off by hardware.

Note that the analog input values must not be allowed to exceed the value of the selected reference voltage.

SAVRS[1:0]	Reference Source	Description
00	V_{DD}	Internal A/D converter power supply
01	VREF Pin	External A/D converter reference pin
1x	V_{VR}	Internal A/D converter PGA output voltage

A/D Converter Reference Voltage Selection

A/D Converter Input Signals

All of the external A/D analog input pins are pin-shared with the I/O pins as well as other functions. The corresponding pin-shared function selection bits in the PxS1 and PxS0 registers, determine whether the external input pins are setup as A/D converter analog channel inputs or whether they have other functions. If the corresponding pin is setup to be an A/D converter analog channel input,

the original pin function will be disabled. In this way, pins can be changed under program control to change their function between A/D inputs and other functions. All pull-high resistors, which are setup through register programming, will be automatically disconnected if the pins are setup as A/D inputs. Note that it is not necessary to first setup the A/D pin as an input in the port control register to enable the A/D input as when the relevant A/D input function selection bits enable an A/D input, the status of the port control register will be overridden.

As the device contains only one actual analog to digital converter hardware circuit, each of the external and internal analog signals must be routed to the converter. The SAINS3~SAINS0 bits in the SADC1 register are used to determine that the analog signal to be converted comes from the external channel input or internal analog signal. The SACS3~SACS0 bits in the SADC0 register are used to determine which external channel input is selected to be converted. If the SAINS3~SAINS0 bits are set to "0000", "0100" or "1100~1111", the external channel input will be selected to be converted and the SACS3~SACS0 bits can determine which external channel is selected.

When the SAINS3~SAINS0 bits are set to the value of "0001~0011", "0101~1000" or "1001~1011", the internal analog signal will be selected. If the internal analog signal is selected to be converted, the external channel signal input will automatically be switched off regardless of the SACS3~SACS0 bit values. It will prevent the external channel input from being connected together with the internal analog signal.

SAINS[3:0]	SACS[3:0]	Input Signals	Description
0000, 0100, 1100~1111	0000~1010	AN0~AN10	External channel analog input ANn
	1011~1111	—	Floating
0001	xxxx	V_{DD}	Internal A/D converter power supply voltage V_{DD}
0010	xxxx	$V_{DD}/2$	Internal A/D converter power supply voltage $V_{DD}/2$
0011	xxxx	$V_{DD}/4$	Internal A/D converter power supply voltage $V_{DD}/4$
0101	xxxx	V_{VR}	Internal A/D converter PGA output V_{VR}
0110	xxxx	$V_{VR}/2$	Internal A/D converter PGA output $V_{VR}/2$
0111	xxxx	$V_{VR}/4$	Internal A/D converter PGA output $V_{VR}/4$
1000	xxxx	V_{OPAO}	Internal operational amplifier output voltage
1001~1011	xxxx	V_{SS}	Connected to the ground

A/D Converter Input Signal Selection

A/D Conversion Operation

The START bit in the SADC0 register is used to start the AD conversion. When the microcontroller sets this bit from low to high and then low again, an analog to digital conversion cycle will be initiated.

The ADBZ bit in the SADC0 register is used to indicate whether the analog to digital conversion process is in progress or not. This bit will be automatically set to 1 by the microcontroller after an A/D conversion is successfully initiated. When the A/D conversion is complete, the ADBZ bit will be cleared to 0. In addition, the corresponding A/D interrupt request flag will be set in the interrupt control register, and if the interrupts are enabled, an internal interrupt signal will be generated. This A/D internal interrupt signal will direct the program flow to the associated A/D internal interrupt address for processing. If the A/D internal interrupt is disabled, the microcontroller can poll the ADBZ bit in the SADC0 register to check whether it has been cleared as an alternative method of detecting the end of an A/D conversion cycle.

The clock source for the A/D converter, which originates from the system clock f_{SYS}, can be chosen to be either f_{SYS} or a subdivided version of f_{SYS}. The division ratio value is determined by the SACKS2~SACKS0 bits in the SADC1 register. Although the A/D clock source is determined by the system clock f_{SYS} and by bits SACKS2~SACKS0, there are some limitations on the maximum A/D clock source speed that can be selected. As the recommended range of permissible A/D clock period,

t_{ADCK}, is from 0.5μs to 10μs, care must be taken for system clock frequencies. For example, if the system clock operates at a frequency of 8MHz, the SACKS2~SACKS0 bits should not be set to 000, 001 or 111. Doing so will give A/D clock periods that are less than the minimum A/D clock period or greater than the maximum A/D clock period which may result in inaccurate A/D conversion values. Refer to the following table for examples, where values marked with an asterisk * special care must be taken.

f_{SYS}	A/D Clock Period (t_{ADCK})							
	SACKS[2:0] =000 (f_{SYS})	SACKS[2:0] =001 (f_{SYS}/2)	SACKS[2:0] =010 (f_{SYS}/4)	SACKS[2:0] =011 (f_{SYS}/8)	SACKS[2:0] =100 (f_{SYS}/16)	SACKS[2:0] =101 (f_{SYS}/32)	SACKS[2:0] =110 (f_{SYS}/64)	SACKS[2:0] =111 (f_{SYS}/128)
1MHz	1μs	2μs	4μs	8μs	16μs *	32μs *	64μs *	128μs *
2MHz	500ns	1μs	2μs	4μs	8μs	16μs *	32μs *	64μs *
4MHz	250ns *	500ns	1μs	2μs	4μs	8μs	16μs *	32μs *
8MHz	125ns *	250ns *	500ns	1μs	2μs	4μs	8μs	16μs *
12MHz	83ns *	167ns *	333ns *	667ns	1.33μs	2.67μs	5.33μs	10.67μs *
16MHz	62.5ns *	125ns *	250ns *	500ns	1μs	2μs	4μs	8μs

A/D Clock Period Examples

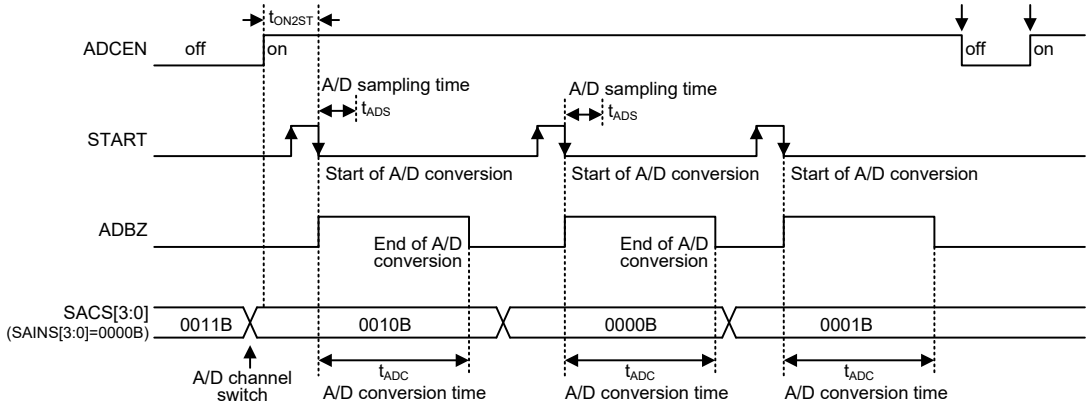
Controlling the power on/off function of the A/D converter circuitry is implemented using the ADCEN bit in the SADC0 register. This bit must be set high to power on the A/D converter. When the ADCEN bit is set high to power on the A/D converter internal circuitry, a certain delay as indicated in the timing diagram must be allowed before an A/D conversion is initiated. Even if no pins are selected for use as A/D inputs, if the ADCEN bit is high, then some power will still be consumed. In power conscious applications it is therefore recommended that the ADCEN is cleared to zero to reduce power consumption when the A/D converter function is not being used.

Conversion Rate and Timing Diagram

A complete A/D conversion contains two parts, data sampling and data conversion. The data sampling which is defined as t_{ADS} takes 4 A/D clock periods and the data conversion takes 12 A/D clock periods. Therefore a total of 16 A/D clock periods for an analog signal A/D conversion which is defined as t_{ADC} are necessary.

Maximum single A/D conversion rate = 1/(A/D clock period × 16)

The accompanying diagram shows graphically the various stages involved in an external channel input signal analog to digital conversion process and its associated timing. After an A/D conversion process has been initiated by the application program, the microcontroller internal hardware will begin to carry out the conversion, during which time the program can continue with other functions. The time taken for the A/D conversion is 16 t_{ADCK} where t_{ADCK} is equal to the A/D clock period.



A/D Conversion Timing – External Channel Input

Summary of A/D Conversion Steps

The following summarises the individual steps that should be executed in order to implement an A/D conversion process.

- Step 1

  Select the required A/D conversion clock by properly programming the SACKS2~SACKS0 bits in the SADC1 register.

- Step 2

  Enable the A/D converter by setting the ADCEN bit in the SADC0 register to one.

- Step 3

  Select which signal is to be connected to the internal A/D converter by correctly configuring the SAINS3~SAINS0 and SACS3~SACS0 bits.

  Selecting the external channel input to be converted, go to Step 4.

  Selecting the internal analog signal to be converted, go to Step 5.

- Step 4

  If the SAINS3~SAINS0 bits are set to "0000", "0100" or "1100~1111", the external channel input can be selected. The desired external channel input is selected by configuring the SACS3~SACS0 bits. When the A/D input signal comes from the external channel input, the corresponding pin should be configured as an A/D input function by selecting the relevant pin-shared function control bits. Then go to Step 6.

- Step 5

  If the SAINS3~SAINS0 bits are set to "0001~0011", "0101~1000" or "1001~1011", the relevant internal analog signal will be selected. When the internal analog signal is selected to be converted, the external channel analog input will automatically be disconnected. Then go to Step 6.

- Step 6

  Select the A/D converter output data format by configuring the ADRFS bit.

- Step 7

  Select the A/D converter reference voltage source by configuring the SAVRS1~SAVRS0 bits.

  Select the PGA input signal and the desired PGA gain if the PGA output voltage, V_{VR}, is selected as the A/D converter reference voltage.

- Step 8

  If A/D conversion interrupt is used, the interrupt control registers must be correctly configured to ensure the A/D interrupt function is active. The master interrupt control bit, EMI, and the A/D conversion interrupt control bit, ADE, must both be set high in advance.

- Step 9

  The A/D conversion procedure can now be initialized by setting the START bit from low to high and then low again.

- Step 10

  If A/D conversion is in progress, the ADBZ flag will be set high. After the A/D conversion process is complete, the ADBZ flag will go low and then the output data can be read from SADOH and SADOL registers.

Note: When checking for the end of the conversion process, if the method of polling the ADBZ bit in the SADC0 register is used, the interrupt enable step above can be omitted.

Programming Considerations

During microcontroller operations where the A/D converter is not being used, the A/D internal circuitry can be switched off to reduce power consumption, by setting bit ADCEN low in the SADC0 register. When this happens, the internal A/D converter circuits will not consume power irrespective of what analog voltage is applied to their input lines. If the A/D converter input lines are used as normal I/Os, then care must be taken as if the input voltage is not at a valid logic level, then this may lead to some increase in power consumption.

A/D Transfer Function

As the device contains a 12-bit A/D converter, its full-scale converted digitised value is equal to FFFH. Since the full-scale analog input value is equal to the actual A/D converter reference voltage, V_{REF}, this gives a single bit analog input value of reference voltage value divided by 4096.

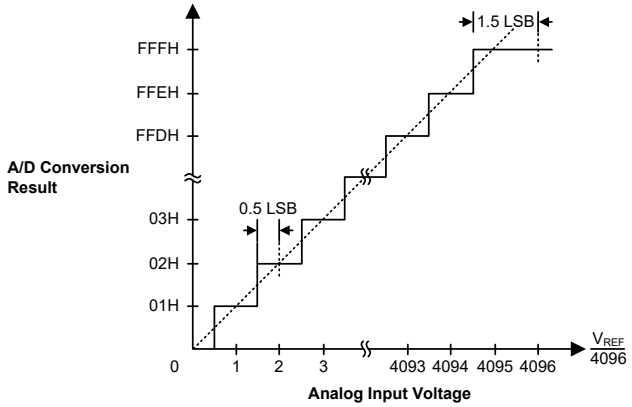
$$1\ LSB = V_{REF}/4096$$

The A/D Converter input voltage value can be calculated using the following equation:

$$\text{A/D input voltage} = \text{A/D output digital value} \times V_{REF} / 4096$$

The diagram shows the ideal transfer function between the analog input value and the digitised output value for the A/D converter. Except for the digitised zero value, the subsequent digitised values will change at a point 0.5 LSB below where they would change without the offset, and the last full scale digitised value will change at a point 1.5 LSB below the V_{REF} level.

Note that here the V_{REF} voltage is the actual A/D converter reference voltage determined by the SAVRS1~SAVRS0 bits.



Ideal A/D Transfer Function

A/D Programming Examples

The following two programming examples illustrate how to setup and implement an A/D conversion. In the first example, the method of polling the ADBZ bit in the SADC0 register is used to detect when the conversion cycle is complete, whereas in the second example, the A/D interrupt is used to determine when the conversion is complete.

Example: using an ADBZ polling method to detect the end of conversion

```
clr ADE              ; disable ADC interrupt
mov a,03H            ; select fSYS/8 as A/D clock and A/D input
mov SADC1,a          ; signal comes from external channel
mov a,00H            ; select VDD as the A/D reference voltage source
mov SADC2,a
mov a,30H            ; setup PAS1 to configure pin AN0
mov PAS1,a
```

```
mov a,20H            ; enable A/D converter and select AN0 as
mov SADC0,a          ; the A/D external channel input
:
start_conversion:
clr START            ; high pulse on start bit to initiate conversion
set START            ; reset A/D
clr START            ; start A/D
:
polling_EOC:
sz  ADBZ             ; poll the SADC0 register ADBZ bit to detect end of A/D
conversion
jmp polling_EOC      ; continue polling
:
mov a,SADOL          ; read low byte conversion result value
mov SADOL_buffer,a   ; save result to user defined register
mov a,SADOH          ; read high byte conversion result value
mov SADOH_buffer,a   ; save result to user defined register
:
jmp start_conversion ; start next A/D conversion
```

Example: using the interrupt method to detect the end of conversion

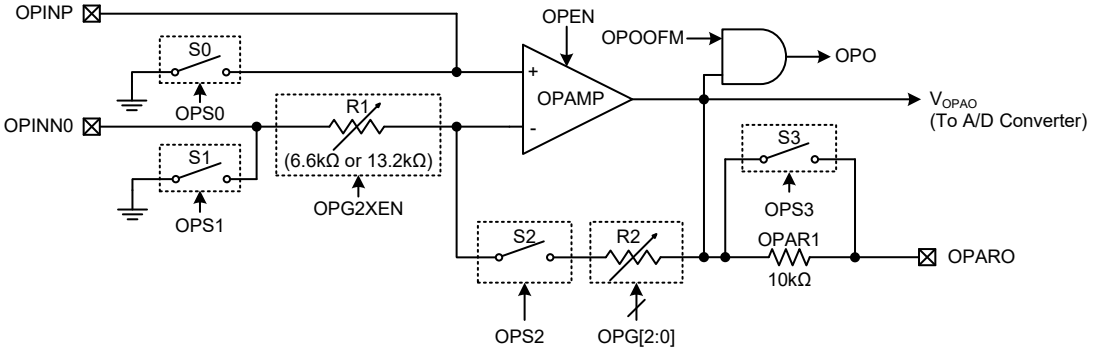
```
clr ADE              ; disable ADC interrupt
mov a,03H            ; select fSYS/8 as A/D clock and A/D input
mov SADC1,a          ; signal comes from external channel
mov a,00H            ; select VDD as the A/D reference voltage source
mov SADC2,a
mov a,30H            ; setup PAS1 to configure pin AN0
mov PAS1,a
mov a,20H            ; enable A/D converter and select AN0 as
mov SADC0,a          ; the A/D external channel input
:
start_conversion:
clr START            ; high pulse on START bit to initiate conversion
set START            ; reset A/D
clr START            ; start A/D
clr ADF              ; clear ADC interrupt request flag
set ADE              ; enable ADC interrupt
set EMI              ; enable global interrupt
:
:
ADC_ISR:             ; ADC interrupt service routine
mov acc_stack,a      ; save ACC to user defined memory
mov a,STATUS
mov status_stack,a   ; save STATUS to user defined memory
:
mov a,SADOL          ; read low byte conversion result value
mov SADOL_buffer,a   ; save result to user defined register
mov a,SADOH          ; read high byte conversion result value
mov SADOH_buffer,a   ; save result to user defined register
:
EXIT_INT_ISR:
mov a,status_stack
mov STATUS,a         ; restore STATUS from user defined memory
mov a,acc_stack      ; restore ACC from user defined memory
reti
```

Operational Amplifier

The device includes an operational amplifier for measure applications. An operational amplifier, OPAMP, produces an output potential that is typically hundreds of thousands of times larger than the potential difference between its input terminals. By integrating the OPAMP electronic circuitry into the microcontroller, the need for external components is reduced greatly.

Operational Amplifier Operation

The Operational Amplifier can be used for signal amplification according to specific user requirements. The gain is selectable by using the OPG[2:0] bit field together with the OPG2XEN bit. The amplified output can be directly output on the OPARO pin, and also be internally connected to the A/D converter for the amplified input signal read.



Note: The OPINP and OPINN0 are pin-shared with the same I/O pin, therefore the positive and negative inputs cannot be used at the same time.

Operational Amplifier Block Diagram

Operational Amplifier Registers

The internal Operational Amplifier normal operation and input offset voltage calibration function are controlled by three registers. The OPC0 register is used to control the switches on or off. The OPC1 register is used to control the OPAMP function enable or disable and select the gain. The OPO bit together with the OPOCAL register are used in the offset calibration procedure.

Register Name	Bit							
	7	6	5	4	3	2	1	0
OPC0	—	—	—	—	OPS3	OPS2	OPS1	OPS0
OPC1	OPEN	—	—	OPO	OPG2XEN	OPG2	OPG1	OPG0
OPOCAL	OPOOFM	OPORSP	OPOOF5	OPOOF4	OPOOF3	OPOOF2	OPOOF1	OPOOF0

Operational Amplifier Register List

• OPC0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	OPS3	OPS2	OPS1	OPS0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3 **OPS3:** OPAMP switch S3 on/off control
0: Off
1: On

Bit 2 **OPS2:** OPAMP switch S2 on/off control
0: Off
1: On

Bit 1 **OPS1:** OPAMP switch S1 on/off control
0: Off
1: On

Bit 0 **OPS0:** OPAMP switch S0 on/off control
0: Off
1: On

• OPC1 Register

Bit	7	6	5	4	3	2	1	0
Name	OPEN	—	—	OPO	OPG2XEN	OPG2	OPG1	OPG0
R/W	R/W	—	—	R	R/W	R/W	R/W	R/W
POR	0	—	—	0	0	0	0	0

Bit 7 **OPEN:** OPAMP function enable or disable control
0: Disable
1: Enable

Bit 6~5 Unimplemented, read as "0"

Bit 4 **OPO:** OPAMP output status (positive logic)
This bit is read only. When the OPOOFM bit is set to 1, the OPO is defined as OPAMP output status, refer to the Input Offset Calibration section.

Bit 3 **OPG2XEN:** R2/R1 ratio doubling enable control
0: Disable – Select R1=13.2kΩ
1: Enable – Select R1=6.6kΩ
When this bit is set to 1, the R2/R1 ratio selected by the OPG2~OPG0 bits will be doubled.

Bit 2~0 **OPG2~OPG0:** R2/R1 ratio selection
If OPG2XEN=0, R1=13.2kΩ:
000: R2/R1=2
001: R2/R1=5
010: R2/R1=6
011: R2/R1=7
100: R2/R1=20
101: R2/R1=25
110: R2/R1=30
111: R2/R1=35
If OPG2XEN=1, R1=6.6kΩ:
The R2/R1 ratio doubling function is enabled, the above R2/R1 value will be doubled.
Note that the internal resistors, R1 and R2, should be used when the gain is determined by these bits. This means the S1 switch should be on or the OPINN0 should be selected together with the S2 switch on. Otherwise, the gain accuracy will not be guaranteed.

• OPOCAL Register

Bit	7	6	5	4	3	2	1	0
Name	OPOOFM	OPORSP	OPOOF5	OPOOF4	OPOOF3	OPOOF2	OPOOF1	OPOOF0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	1	0	0	0	0	0

Bit 7 **OPOOFM:** OPAMP operating mode selection
0: Normal operation mode
1: Offset calibration mode

Bit 6 **OPORSP:** OPAMP input offset voltage calibration reference selection
0: Reserved
1: Select non-inverting input as the reference voltage input
Note that this bit must be set to "1" after power on.

Bit 5~0 **OPOOF5~OPOOF0:** OPAMP input offset voltage calibration control
This 6-bit field is used to perform the OPAMP input offset calibration operation and the value for the OPAMP input offset calibration can be restored into this bit field. More detailed information is described in the "Input Offset Calibration" section.

Input Voltage Range

Together with different PGA operating modes, the input voltage on the OPAMP pins can be positive or negative for flexible application. There are two operating mode examples below. In these examples the internal resistors, R1 and R2, are used respectively.

- For $V_{IN} > 0$, the PGA operates in the non-inverting mode and the output voltage of the PGA is obtained using the formula below:

$$VO_{PGA}=(1+R2/R1)\times V_{IN}$$

- When the S0 and S2 switches are ON, S1 and S3 switches are OFF and the input node is OPINN0. For $0 > V_{IN} > -0.2V$, the PGA operates in the inverting mode, the output voltage of the PGA is obtained using the formula below:

$$VO_{PGA}= -(R2/R1)\times V_{IN}$$

Note that if V_{IN} is negative, it cannot be lower than -0.2V which will result in current leakage.

Input Offset Calibration

This OPAMP includes an input offset calibration function. The calibrated data is stored in the OPOOF[5:0] bits. The OPOOFM bit is the calibration mode control bit and the OPORSP bit is used to indicate that the input reference voltage comes from OPINP in the calibration mode. The OPINP pin is the OPAMP positive input pin and the OPARO pin is the OPAMP analog output pin. The OPAMP digital output flag is OPO, which is used for OPAMP calibration mode.

Offset Calibration Procedure

Note that as the OPAMP input pins are pin-shared with other functions, they should be configured as OPAMP inputs first by properly configuring the related pin-shared control bits. For operational amplifier input offset calibration, the procedures are summarized as follows.

Step 1: Set OPOOFM=1 and OPORSP=1, the OPAMP is now under offset calibration mode. To make sure V_{OS} as minimal as possible after calibration, the input reference voltage in calibration should be the same as input DC operating voltage in normal mode operation.

Step 2: Set OPOOF[5:0]=000000 and then read the OPO bit.

Step 3: Increase the OPOOF[5:0] value by 1 and then read the OPO bit.
If the OPO bit state has not changed, then repeat Step 3 until the OPO bit state has changed.
If the OPO bit state has changed, record the OPOOF[5:0] value as V_{OS1} and then go to Step 4.

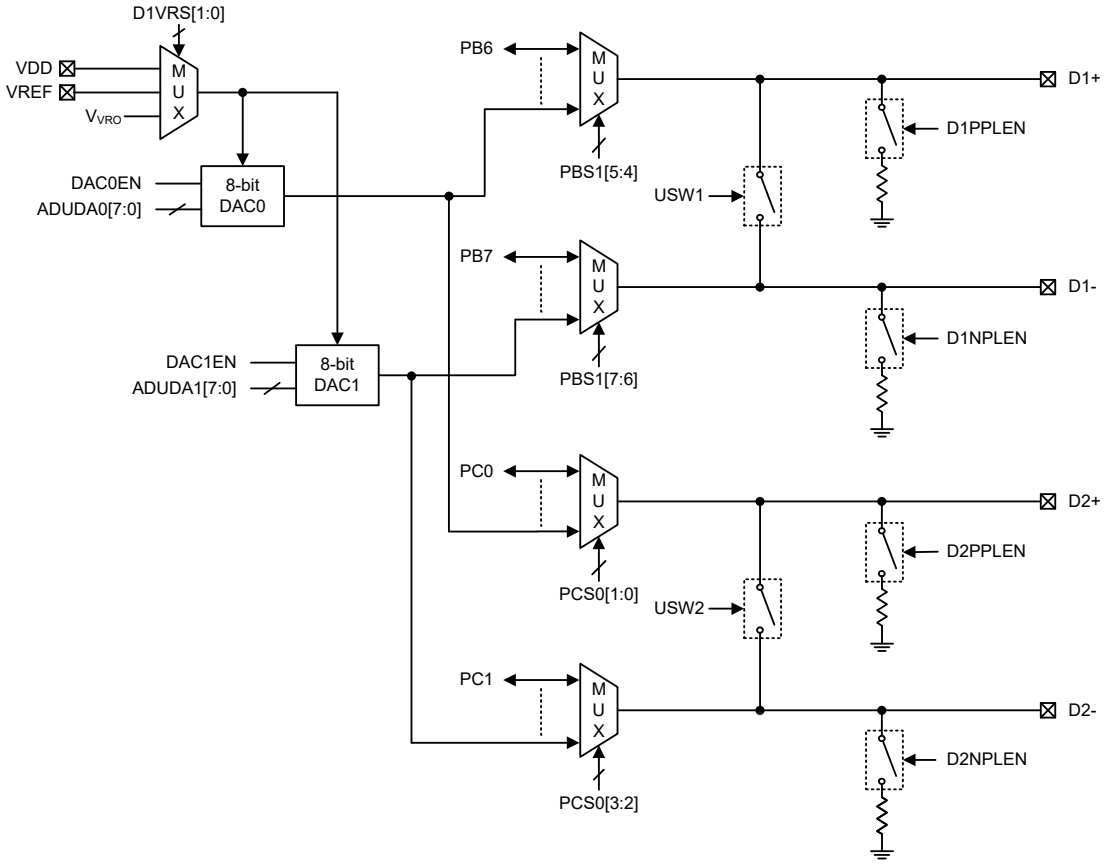
Step 4: Set OPOOF[5:0]=111111 then read the OPO bit.

Step 5: Decrease the OPOOF[5:0] value by 1 and then read the OPO bit.
If the OPO bit state has not changed, then repeat Step 5 until the OPO bit state has changed.
If the OPO bit state has changed, record the OPOOF[5:0] value as V_{OS2} and then go to Step 6.

Step 6: Restore the Operational Amplifier input offset calibration value V_{OS} into the OPOOF[5:0] bit field. The offset Calibration procedure is now finished.
Where $V_{OS}=(V_{OS1}+V_{OS2})/2$. If $(V_{OS1}+V_{OS2})/2$ is not integral, discard the decimal.
Residue $V_{OS}=V_{OUT}-V_{IN}$

USB Auto Detection

The device includes two USB ports named D1+/D1- and D2+/D2- to implement the Charge/ Discharge Auto Detection function. Users can distinguish the device connected to the USB ports is a dedicated charger, portable device, general USB interface or charging device with USB interface by monitoring the voltage and current of the connected USB lines.



USB Auto Detection Block Diagram

D1+/D1- and D2+/D2- for Auto Detection

In the USB auto detection circuit there are two 8-bit D/A Converters, DACn, which is enabled by setting the DACnEN bit in the ADUC0 register. The D/A Converter output signal is controlled by the ADUDAn register value and the reference voltage which is selected by the D1VRS[1:0] bits in the ADUC0 register. There is an analog switch connected between the D1+ and D1- lines, which is controlled by the USW1 bit. Similarly, there is an analog switch connected between the D2+ and D2- lines, which is controlled by the USW2 bit. However it needs to note that only when one of the D1+ and D1- or D2+ and D2- pins is in the analog or digital input type by setting the pin-shared function register, and then set the USW1 or USW2 bit high, can the switch be on. The D1+/D1- and D2+/D2- lines are individually connected a pull-low resistor to VSS which are controlled by the D1PPLEN/D1NPLEN and D2PPLEN/D2NPLEN bits in the ADUC1 register respectively.

USB Auto Detection Registers

Overall operation of the USB auto detection function is controlled using several registers.

Register Name	Bit							
	7	6	5	4	3	2	1	0
ADUC0	—	—	D1VRS1	D1VRS0	—	—	DAC1EN	DAC0EN
ADUC1	—	—	D2NPLEN	D2PPLEN	D1NPLEN	D1PPLEN	USW2	USW1
ADUDA0	D7	D6	D5	D4	D3	D2	D1	D0
ADUDA1	D7	D6	D5	D4	D3	D2	D1	D0

USB Auto Detection Register List

• ADUC0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	D1VRS1	D1VRS0	—	—	DAC1EN	DAC0EN
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5~4 **D1VRS1~ D1VRS0**: DAC1 and DAC0 reference voltage selection
00/11: From VDD pin
01: From VREF pin
10: From V_{VRO}

Bit 3~2 Unimplemented, read as "0"

Bit 1 **DAC1EN:** DAC1 enable control
0: Disable
1: Enable

Bit 0 **DAC0EN:** DAC0 enable control
0: Disable
1: Enable

• ADUC1 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	D2NPLEN	D2PPLEN	D1NPLEN	D1PPLEN	USW2	USW1
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **D2NPLEN:** D2- pin pull-low control
0: Disable
1: Enable

Bit 4 **D2PPLEN:** D2+ pin pull-low control
0: Disable
1: Enable

Bit 3 **D1NPLEN:** D1- pin pull-low control
0: Disable
1: Enable

Bit 2 **D1PPLEN:** D1+ pin pull-low control
0: Disable
1: Enable

Bit 1 **USW2:** USW2 switch on/off control
0: Off
1: On
This bit controls the USW2 switch on/off. However only when one of the D2+ and D2- pins is used as the analog or digital input pin by setting the pin-shared function register, and then set the USW2 bit high, can the switch be on.

Bit 0 **USW1:** USW1 switch on/off control
0: Off
1: On
This bit controls the USW1 switch on/off. However only when one of the D1+ and D1- pins is used as the analog or digital input pin by setting the pin-shared function register, and then set the USW1 bit high, can the switch be on.

• ADUDA0 Register

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0:** 8-bit DAC0 output control data bits
DAC0 output = DAC0 reference voltage × ADUDA0[7:0] / 256

• ADUDA1 Register

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0:** 8-bit DAC1 output control data bits
DAC1 output = DAC1 reference voltage × ADUDA1[7:0] / 256

Serial Interface Module – SIM

The device contains a Serial Interface Module, which includes both the four-line SPI interface or two-line I^2C interface types, to allow an easy method of communication with external peripheral hardware. Having relatively simple communication protocols, these serial interface types allow the microcontroller to interface to external SPI or I^2C based hardware such as sensors, Flash or EEPROM memory, etc. The SIM interface pins are pin-shared with other I/O pins and therefore the SIM interface functional pins must first be selected using the corresponding pin-shared function selection bits. As both interface types share the same pins and registers, the choice of whether the SPI or I^2C type is used is made using the SIM operating mode control bits, named SIM2~SIM0, in the SIMC0 register. These pull-high resistors of the SIM pin-shared I/O pins are selected using pull-high control registers when the SIM function is enabled and the corresponding pins are used as SIM input pins.

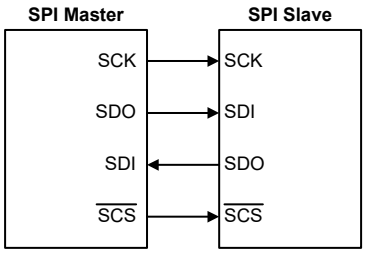
SPI Interface

The SPI interface is often used to communicate with external peripheral devices such as sensors, Flash or EEPROM memory devices, etc. Originally developed by Motorola, the four line SPI interface is a synchronous serial data interface that has a relatively simple communication protocol simplifying the programming requirements when communicating with external hardware devices.

The communication is full duplex and operates as a slave/master type, where the device can be either master or slave. Although the SPI interface specification can control multiple slave devices from a single master, the device is provided only one $\overline{SCS}$ pin. If the master needs to control multiple slave devices from a single master, the master can use I/O pin to select the slave devices.

SPI Interface Operation

The SPI interface is a full duplex synchronous serial data link. It is a four line interface with pin names SDI, SDO, SCK and $\overline{SCS}$. Pins SDI and SDO are the Serial Data Input and Serial Data Output lines, SCK is the Serial Clock line and $\overline{SCS}$ is the Slave Select line. As the SPI interface pins are pin-shared with normal I/O pins and with the I^2C function pins, the SPI interface pins must first be selected by configuring the pin-shared function selection bits and setting the correct bits in the SIMC0 and SIMC2 registers. After the desired SPI configuration has been set it can be disabled or enabled using the SIMEN bit in the SIMC0 register. Communication between devices connected to the SPI interface is carried out in a slave/master mode with all data transfer initiations being implemented by the master. The Master also controls the clock signal. As the device only contains a single $\overline{SCS}$ pin only one slave device can be utilized. The $\overline{SCS}$ pin is controlled by software, set CSEN bit to 1 to enable $\overline{SCS}$ pin function, set CSEN bit to 0 the $\overline{SCS}$ pin will be floating state.

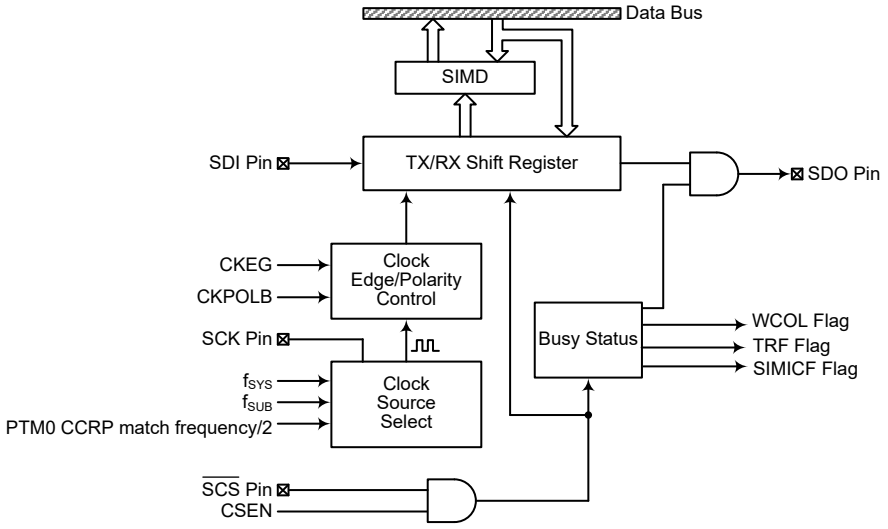


SPI Master/Slave Connection

The SPI function in the device offers the following features:

- Full duplex synchronous data transfer
- Both Master and Slave modes
- LSB first or MSB first data transmission modes
- Transmission complete flag
- Rising or falling active clock edge

The status of the SPI interface pins is determined by a number of factors such as whether the device is in the master or slave mode and upon the condition of certain control bits such as CSEN and SIMEN.



SPI Block Diagram

SPI Registers

There are three internal registers which control the overall operation of the SPI interface. These are the SIMD data register and two registers SIMC0 and SIMC2.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	—	SIMDEB1	SIMDEB0	SIMEN	SIMICF
SIMC2	D7	D6	CKPOLB	CKEG	MLS	CSEN	WCOL	TRF
SIMD	D7	D6	D5	D4	D3	D2	D1	D0

SPI Register List

SPI Data Register

The SIMD register is used to store the data being transmitted and received. The same register is used by both the SPI and I²C functions. Before the device writes data to the SPI bus, the actual data to be transmitted must be placed in the SIMD register. After the data is received from the SPI bus, the device can read it from the SIMD register. Any transmission or reception of data from the SPI bus must be made via the SIMD register.

- **SIMD Register**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	x	x	x	x	x	x	x	x

"x": unknown

Bit 7~0 **D7~D0:** SIM data register bit 7 ~ bit 0

SPI Control Registers

There are also two control registers for the SPI interface, SIMC0 and SIMC2. Note that the SIMC2 register also has the name SIMA which is used by the I²C function. The SIMC1 register is not used by the SPI function, only by the I²C function. Register SIMC0 is used to control the enable/disable function and to set the data transmission clock frequency. Register SIMC2 is used for other control functions such as LSB/MSB selection, write collision flag, etc.

- **SIMC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	SIM2	SIM1	SIM0	—	SIMDEB1	SIMDEB0	SIMEN	SIMICF
R/W	R/W	R/W	R/W	—	R/W	R/W	R/W	R/W
POR	1	1	1	—	0	0	0	0

Bit 7~5 **SIM2~SIM0:** SIM Operating Mode Control
000: SPI master mode; SPI clock is f_{SYS}/4
001: SPI master mode; SPI clock is f_{SYS}/16
010: SPI master mode; SPI clock is f_{SYS}/64
011: SPI master mode; SPI clock is f_{SUB}
100: SPI master mode; SPI clock is PTM0 CCRP match frequency/2
101: SPI slave mode
110: I²C slave mode
111: Non SIM function

These bits setup the overall operating mode of the SIM function. As well as selecting if the I²C or SPI function, they are used to control the SPI Master/Slave selection and the SPI Master clock frequency. The SPI clock is a function of the system clock but can also be chosen to be sourced from PTM0 and f_{SUB}. If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4 Unimplemented, read as "0"

Bit 3~2 **SIMDEB1~SIMDEB0:** I²C Debounce Time Selection

The SIMDEB1~SIMDEB0 bits are only used in the I²C mode and the detailed definition is described in the I²C section.

Bit 1 **SIMEN:** SIM Enable Control

0: Disable
1: Enable

The bit is the overall on/off control for the SIM interface. When the SIMEN bit is cleared to zero to disable the SIM interface, the SDI, SDO, SCK and $\overline{SCS}$, or SDA and SCL lines will lose their SPI or I²C function and the SIM operating current will be reduced to a minimum value. When the bit is high the SIM interface is enabled. If the SIM is configured to operate as an SPI interface via the SIM2~SIM0 bits, the contents of the SPI control registers will remain at the previous settings when the SIMEN bit changes from low to high and should therefore be first initialised by the application program. If the SIM is configured to operate as an I²C interface via the SIM2~SIM0 bits and the SIMEN bit changes from low to high, the contents of the I²C control bits such as HTX and TXAK will remain at the previous settings and should therefore be first initialised by the application program while the relevant I²C flags such as HCF, HAAS, HBB, SRW and RXAK will be set to their default states.

Bit 0 **SIMICF:** SIM SPI slave mode Incomplete Transfer Flag

0: SIM SPI slave mode incomplete condition not occurred
1: SIM SPI slave mode incomplete condition occurred

This bit is only available when the SIM is configured to operate in an SPI slave mode. If the SPI operates in the slave mode with the SIMEN and CSEN bits both being set to 1 but the $\overline{SCS}$ line is pulled high by the external master device before the SPI data transfer is completely finished, the SIMICF bit will be set to 1 together with the TRF bit. When this condition occurs, the corresponding interrupt will occur if the interrupt function is enabled. However, the TRF bit will not be set to 1 if the SIMICF bit is set to 1 by software application program.

• SIMC2 Register

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	CKPOLB	CKEG	MLS	CSEN	WCOL	TRF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 Undefined bits

These bits can be read or written by the application program.

Bit 5 **CKPOLB:** SPI clock line base condition selection

0: The SCK line will be high when the clock is inactive.
1: The SCK line will be low when the clock is inactive.

The CKPOLB bit determines the base condition of the clock line, if the bit is high, then the SCK line will be low when the clock is inactive. When the CKPOLB bit is low, then the SCK line will be high when the clock is inactive.

Bit 4 **CKEG:** SPI SCK clock active edge type selection

CKPOLB=0

0: SCK is high base level and data capture at SCK rising edge
1: SCK is high base level and data capture at SCK falling edge

CKPOLB=1

0: SCK is low base level and data capture at SCK falling edge
1: SCK is low base level and data capture at SCK rising edge

The CKEG and CKPOLB bits are used to setup the way that the clock signal outputs and inputs data on the SPI bus. These two bits must be configured before data transfer is executed otherwise an erroneous clock edge may be generated. The CKPOLB bit determines the base condition of the clock line, if the bit is high, then the SCK line will be low when the clock is inactive. When the CKPOLB bit is low, then the SCK line will be high when the clock is inactive. The CKEG bit determines active clock edge type which depends upon the condition of CKPOLB bit.

Bit 3 **MLS:** SPI data shift order
0: LSB first
1: MSB first
This is the data shift select bit and is used to select how the data is transferred, either MSB or LSB first. Setting the bit high will select MSB first and low for LSB first.

Bit 2 **CSEN:** SPI $\overline{\text{SCS}}$ pin control
0: Disable
1: Enable
The CSEN bit is used as an enable/disable for the $\overline{\text{SCS}}$ pin. If this bit is low, then the $\overline{\text{SCS}}$ pin will be disabled and placed into a floating condition. If the bit is high, the $\overline{\text{SCS}}$ pin will be enabled and used as a select pin.

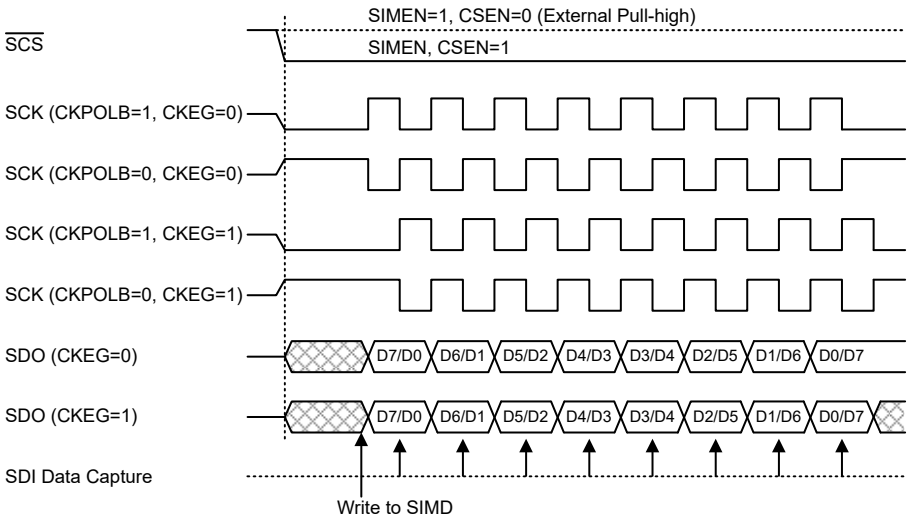
Bit 1 **WCOL:** SPI write collision flag
0: No collision
1: Collision
The WCOL flag is used to detect whether a data collision has occurred or not. If this bit is high, it means that data has been attempted to be written to the SIMD register during a data transfer operation. This writing operation will be ignored if data is being transferred. This bit can be cleared by the application program.

Bit 0 **TRF:** SPI Transmit/Receive complete flag
0: SPI data is being transferred
1: SPI data transfer is completed
The TRF bit is the Transmit/Receive Complete flag and is set to 1 automatically when an SPI data transfer is completed, but must cleared to 0 by the application program. It can be used to generate an interrupt.

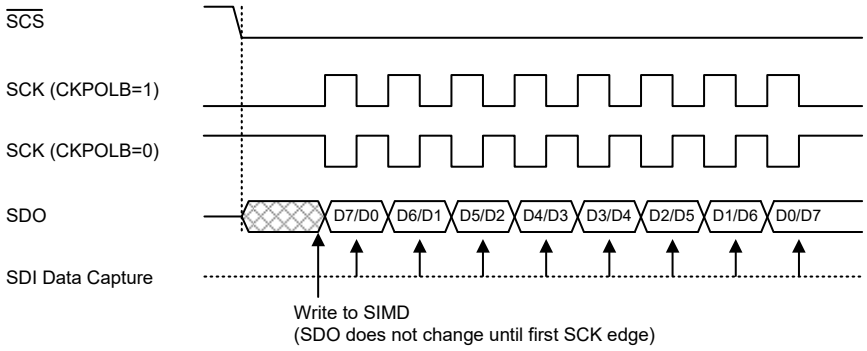
SPI Communication

After the SPI interface is enabled by setting the SIMEN bit high, then in the Master Mode, when data is written to the SIMD register, transmission/reception will begin simultaneously. When the data transfer is complete, the TRF flag will be set automatically, but must be cleared using the application program. In the Slave Mode, when the clock signal from the master has been received, any data in the SIMD register will be transmitted and any data on the SDI pin will be shifted into the SIMD register. The master should output a $\overline{\text{SCS}}$ signal to enable the slave devices before a clock signal is provided. The slave data to be transferred should be well prepared at the appropriate moment relative to the SCK signal depending upon the configurations of the CKPOLB bit and CKEG bit. The accompanying timing diagram shows the relationship between the slave data and SCK signal for various configurations of the CKPOLB and CKEG bits.

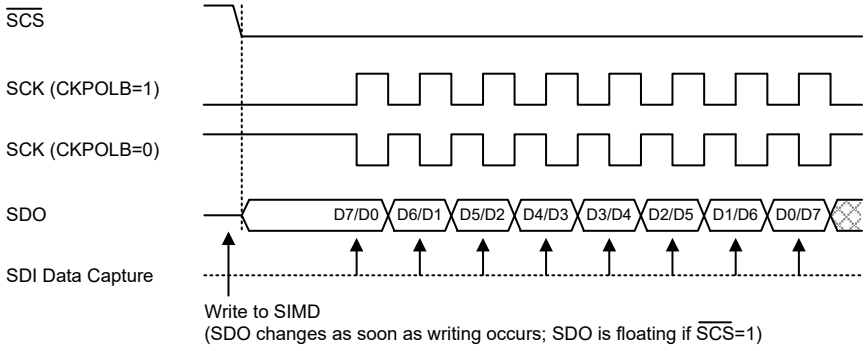
The SPI Master mode will continue to function if the SPI clock is running.



SPI Master Mode Timing

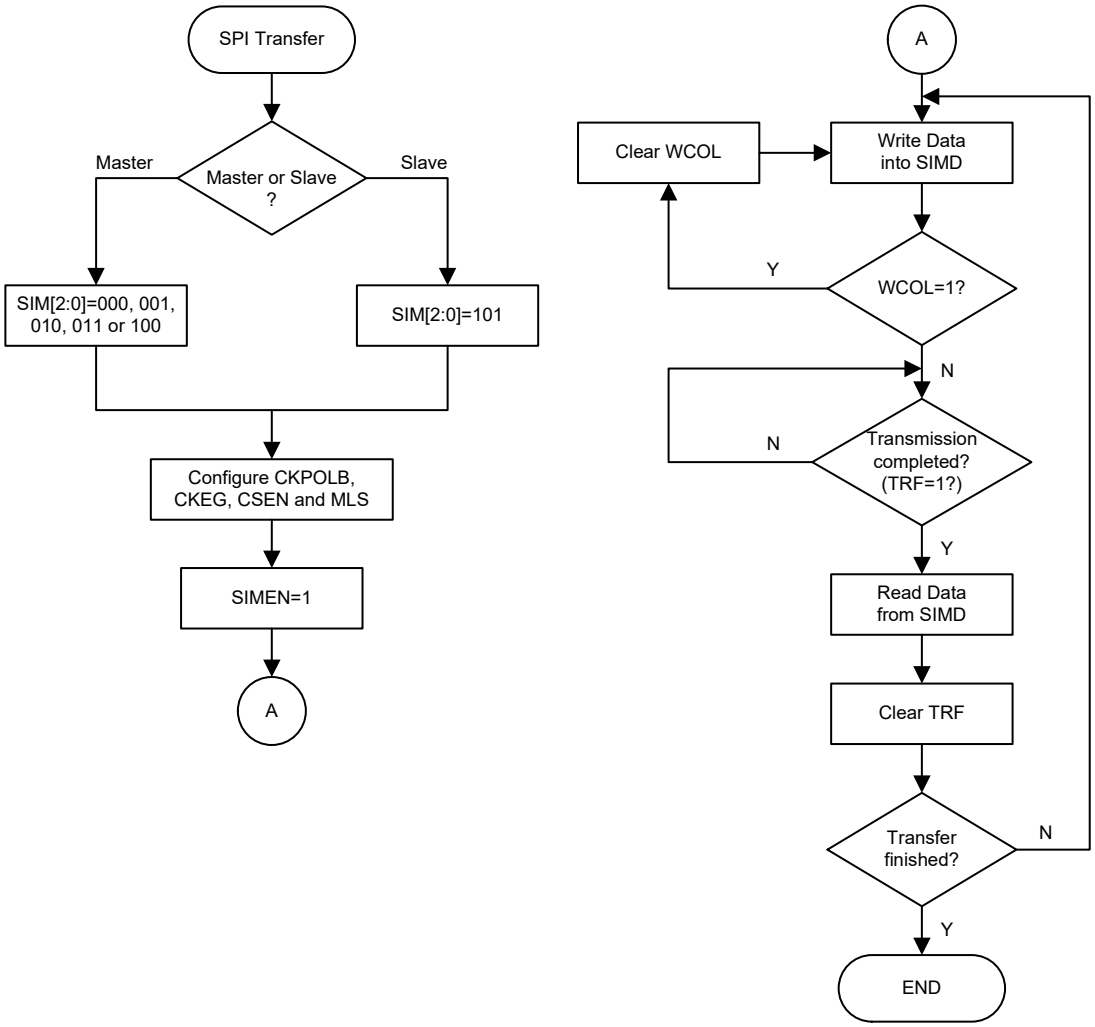


SPI Slave Mode Timing – CKEG=0



Note: For SPI slave mode, if SIMEN=1 and CSEN=0, SPI is always enabled and ignores the $\overline{SCS}$ level.

SPI Slave Mode Timing – CKEG=1



SPI Transfer Control Flowchart

SPI Bus Enable/Disable

To enable the SPI bus, set CSEN=1 and $\overline{SCS}$=0, then wait for data to be written into the SIMD (TXRX buffer) register. For the Master Mode, after data has been written to the SIMD (TXRX buffer) register, then transmission or reception will start automatically. When all the data has been transferred, the TRF bit should be set. For the Slave Mode, when clock pulses are received on SCK, data in the TXRX buffer will be shifted out or data on SDI will be shifted in.

When the SPI bus is disabled, the SCK, SDI, SDO and $\overline{SCS}$ can become I/O pins or other pin-shared functions using the corresponding pin-shared control bits.

SPI Operation Steps

All communication is carried out using the 4-line interface for either Master or Slave Mode.

The CSEN bit in the SIMC2 register controls the overall function of the SPI interface. Setting this bit high will enable the SPI interface by allowing the $\overline{SCS}$ line to be active, which can then be used to control the SPI interface. If the CSEN bit is low, the SPI interface will be disabled and the $\overline{SCS}$ line will be in a floating condition and can therefore not be used for control of the SPI interface. If the CSEN bit and the SIMEN bit in the SIMC0 are set high, this will place the SDI line in a

floating condition and the SDO line high. If in Master Mode the SCK line will be either high or low depending upon the clock polarity selection bit CKPOLB in the SIMC2 register. If in Slave Mode the SCK line will be in a floating condition. If the SIMEN bit is low, then the bus will be disabled and the $\overline{SCS}$, SDI, SDO and SCK will all become I/O pins or the other functions using the corresponding pin-shared control bits. In the Master Mode the Master will always generate the clock signal. The clock and data transmission will be initiated after data has been written into the SIMD register. In the Slave Mode, the clock signal will be received from an external master device for both data transmission and reception. The following sequences show the order to be followed for data transfer in both Master and Slave Mode.

Master Mode

- Step 1

  Select the SPI Master mode and clock source using the SIM2~SIM0 bits in the SIMC0 control register.

- Step 2

  Setup the CSEN bit and setup the MLS bit to choose if the data is MSB or LSB first, this setting must be the same with the Slave devices.

- Step 3

  Setup the SIMEN bit in the SIMC0 control register to enable the SPI interface.

- Step 4

  For write operations: write the data to the SIMD register, which will actually place the data into the TXRX buffer. Then use the SCK and SDO lines to output the data. After this, go to step 5.

  For read operations: the data transferred in on the SDI line will be stored in the TXRX buffer until all the data has been received at which point it will be latched into the SIMD register.

- Step 5

  Check the WCOL bit if set high then a collision error has occurred so return to step 4. If equal to zero then go to the following step.

- Step 6

  Check the TRF bit or wait for a SIM SPI serial bus interrupt.

- Step 7

  Read data from the SIMD register.

- Step 8

  Clear TRF.

- Step 9

  Go to step 4.

Slave Mode

- Step 1

  Select the SPI Slave mode using the SIM2~SIM0 bits in the SIMC0 control register

- Step 2

  Setup the CSEN bit and setup the MLS bit to choose if the data is MSB or LSB first, this setting must be the same with the Master devices.

- Step 3

  Setup the SIMEN bit in the SIMC0 control register to enable the SPI interface.

- Step 4

  For write operations: write the data to the SIMD register, which will actually place the data into the TXRX buffer. Then wait for the master clock SCK and $\overline{SCS}$ signal. After this, go to step 5.

  For read operations: the data transferred in on the SDI line will be stored in the TXRX buffer until all the data has been received at which point it will be latched into the SIMD register.

- Step 5

  Check the WCOL bit if set high then a collision error has occurred so return to step 4. If equal to zero then go to the following step.

- Step 6

  Check the TRF bit or wait for a SIM SPI serial bus interrupt.

- Step 7

  Read data from the SIMD register.

- Step 8

  Clear TRF.

- Step 9

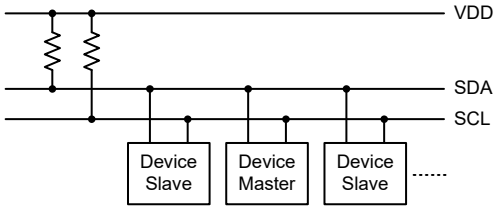
  Go to step 4.

Error Detection

The WCOL bit in the SIMC2 register is provided to indicate errors during data transfer. The bit is set by the SPI serial Interface but must be cleared by the application program. This bit indicates that a data collision has occurred which happens if a write to the SIMD register takes place during a data transfer operation and will prevent the write operation from continuing.

I²C Interface

The I²C interface is used to communicate with external peripheral devices such as sensors, EEPROM memory etc. Originally developed by Philips, it is a two line low speed serial interface for synchronous serial data transfer. The advantage of only two lines for communication, relatively simple communication protocol and the ability to accommodate multiple devices on the same bus has made it an extremely popular interface type for many applications.



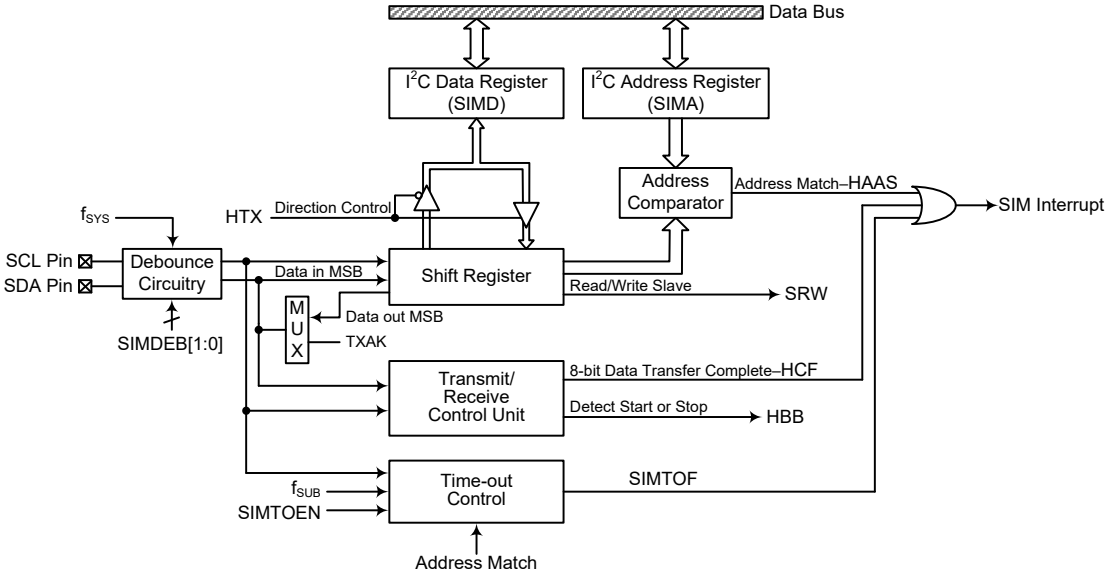
I²C Master/Slave Bus Connection

I²C Interface Operation

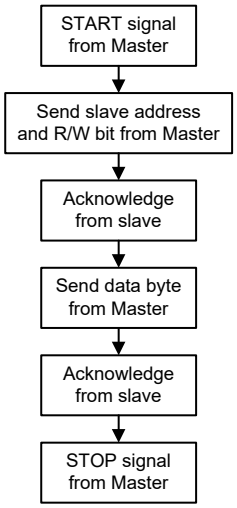
The I²C serial interface is a two line interface, a serial data line, SDA, and serial clock line, SCL. As many devices may be connected together on the same bus, their outputs are both open drain types. For this reason it is necessary that external pull-high resistors are connected to these outputs. Note that no chip select line exists, as each device on the I²C bus is identified by a unique address which will be transmitted and received on the I²C bus.

When two devices communicate with each other on the bidirectional I²C bus, one is known as the master device and one as the slave device. Both master and slave can transmit and receive data;

however, it is the master device that has overall control of the bus. For the device, which only operates in slave mode, there are two methods of transferring data on the I²C bus, the slave transmit mode and the slave receive mode. The pull-high control function pin-shared with SCL/SDA pin is still applicable even if I²C device is activated and the related internal pull-high function could be controlled by its corresponding pull-high control register.



I²C Interface Block Diagram



I²C Interface Operation

The SIMDEB1 and SIMDEB0 bits determine the debounce time of the I²C interface. This uses the system clock to in effect add a debounce time to the external clock to reduce the possibility of glitches on the clock line causing erroneous operation. The debounce time, if selected, can be chosen to be either 2 or 4 system clocks. To achieve the required I²C data transfer speed, there exists a relationship between the system clock, f_{SYS}, and the I²C debounce time. For either the I²C Standard or Fast mode operation, users must take care of the selected system clock frequency and the configured debounce time to match the criterion shown in the following table.

I²C Debounce Time Selection	I²C Standard Mode (100kHz)	I²C Fast Mode (400kHz)
No Debounce	f_{SYS}>2MHz	f_{SYS}>4MHz
2 system clock debounce	f_{SYS}>4MHz	f_{SYS}>8MHz
4 system clock debounce	f_{SYS}>4MHz	f_{SYS}>8MHz

I²C Minimum f_{SYS} Frequency Requirement

I²C Registers

There are three control registers associated with the I²C bus, SIMC0, SIMC1 and SIMTOC, one address register SIMA and one data register, SIMD.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	—	SIMDEB1	SIMDEB0	SIMEN	SIMICF
SIMC1	HCF	HAAS	HBB	HTX	TXAK	SRW	IAMWU	RXAK
SIMA	SIMA6	SIMA5	SIMA4	SIMA3	SIMA2	SIMA1	SIMA0	—
SIMD	D7	D6	D5	D4	D3	D2	D1	D0
SIMTOC	SIMTOEN	SIMTOF	SIMTOS5	SIMTOS4	SIMTOS3	SIMTOS2	SIMTOS1	SIMTOS0

I²C Register List

I²C Data Register

The SIMD register is used to store the data being transmitted and received. The same register is used by both the SPI and I²C functions. Before the device writes data to the I²C bus, the actual data to be transmitted must be placed in the SIMD register. After the data is received from the I²C bus, the device can read it from the SIMD register. Any transmission or reception of data from the I²C bus must be made via the SIMD register.

- **SIMD Register**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	x	x	x	x	x	x	x	x

"x": unknown

Bit 7~0 **D7~D0:** SIM data register bit 7 ~ bit 0

I²C Address Register

The SIMA register is also used by the SPI interface but has the name SIMC2. The SIMA register is the location where the 7-bit slave address of the slave device is stored. Bits 7~1 of the SIMA register define the device slave address. Bit 0 is not implemented.

When a master device, which is connected to the I²C bus, sends out an address, which matches the slave address in the SIMA register, the slave device will be selected. Note that the SIMA register is the same register address as SIMC2 which is used by the SPI interface.

• **SIMA Register**

Bit	7	6	5	4	3	2	1	0
Name	SIMA6	SIMA5	SIMA4	SIMA3	SIMA2	SIMA1	SIMA0	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~1 **SIMA6~SIMA0:** I²C slave address
SIMA6~SIMA0 is the I²C slave address bit 6 ~ bit 0.

Bit 0 **D0:** Reserved bit, can be read or written

I²C Control Registers

There are three control registers for the I²C interface, SIMC0, SIMC1 and SIMTOC. The register SIMC0 is used to control the enable/disable function and to select the I²C slave mode and debounce time. The SIMC1 register contains the relevant flags which are used to indicate the I²C communication status. Another register, SIMTOC, is used to control the I²C time-out function and is described in the corresponding section.

• **SIMC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	SIM2	SIM1	SIM0	—	SIMDEB1	SIMDEB0	SIMEN	SIMICF
R/W	R/W	R/W	R/W	—	R/W	R/W	R/W	R/W
POR	1	1	1	—	0	0	0	0

Bit 7~5 **SIM2~SIM0:** SIM Operating Mode Control
000: SPI master mode; SPI clock is $f_{SYS}/4$
001: SPI master mode; SPI clock is $f_{SYS}/16$
010: SPI master mode; SPI clock is $f_{SYS}/64$
011: SPI master mode; SPI clock is f_{SUB}
100: SPI master mode; SPI clock is PTM0 CCRP match frequency/2
101: SPI slave mode
110: I²C slave mode
111: Non SIM function
These bits setup the overall operating mode of the SIM function. As well as selecting if the I²C or SPI function, they are used to control the SPI Master/Slave selection and the SPI Master clock frequency. The SPI clock is a function of the system clock but can also be chosen to be sourced from PTM0 and f_{SUB}. If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4 Unimplemented, read as "0"

Bit 3~2 **SIMDEB1~SIMDEB0:** I²C Debounce Time Selection
00: No debounce
01: 2 system clock debounce
1x: 4 system clock debounce
These bits are used to select the I²C debounce time when the SIM is configured as the I²C interface function by setting the SIM2~SIM0 bits to "110".

Bit 1 **SIMEN:** SIM Enable Control
0: Disable
1: Enable
The bit is the overall on/off control for the SIM interface. When the SIMEN bit is cleared to zero to disable the SIM interface, the SDI, SDO, SCK and $\overline{SCS}$, or SDA and SCL lines will lose their SPI or I²C function and the SIM operating current will be reduced to a minimum value. When the bit is high the SIM interface is enabled. The SIM configuration option must have first enabled the SIM interface for this bit to be effective. If the SIM is configured to operate as an SPI interface via the SIM2~SIM0 bits, the contents of the SPI control registers will remain at the previous settings when

the SIMEN bit changes from low to high and should therefore be first initialised by the application program. If the SIM is configured to operate as an I²C interface via the SIM2~SIM0 bits and the SIMEN bit changes from low to high, the contents of the I²C control bits such as HTX and TXAK will remain at the previous settings and should therefore be first initialised by the application program while the relevant I²C flags such as HCF, HAAS, HBB, SRW and RXAK will be set to their default states.

Bit 0 **SIMICF:** SIM SPI Incomplete Flag

The SIMICF bit is only used in the SPI mode and the detailed definition is described in the SPI section.

• SIMC1 Register

Bit	7	6	5	4	3	2	1	0
Name	HCF	HAAS	HBB	HTX	TXAK	SRW	IAMWU	RXAK
R/W	R	R	R	R/W	R/W	R	R/W	R
POR	1	0	0	0	0	0	0	1

Bit 7 **HCF:** I²C Bus data transfer completion flag
0: Data is being transferred
1: Completion of an 8-bit data transfer

The HCF flag is the data transfer completion flag. This flag will be zero when data is being transferred. Upon completion of an 8-bit data transfer the flag will go high and an interrupt will be generated.

Bit 6 **HAAS:** I²C Bus data transfer completion flag
0: Not address match
1: Address match

The HAAS flag is the address match flag. This flag is used to determine if the slave device address is the same as the master transmit address. If the addresses match then this bit will be high, if there is no match then the flag will be low.

Bit 5 **HBB:** I²C Bus busy flag
0: I²C Bus is not busy
1: I²C Bus is busy

The HBB flag is the I²C busy flag. This flag will be "1" when the I²C bus is busy which will occur when a START signal is detected. The flag will be cleared to "0" when the bus is free which will occur when a STOP signal is detected.

Bit 4 **HTX:** I²C slave device transmitter/receiver selection
0: Slave device is the receiver
1: Slave device is the transmitter

Bit 3 **TXAK:** I²C bus transmit acknowledge flag
0: Slave sends acknowledge flag
1: Slave does not send acknowledge flag

The TXAK flag is the transmit acknowledge flag. After the slave device has received 8 bits of data, this flag will be transmitted to the bus on the 9th clock from the slave device. The slave device must always set the TXAK bit to "0" before further data is received.

Bit 2 **SRW:** I²C slave read/write flag
0: Slave device should be in receive mode
1: Slave device should be in transmit mode

The SRW flag is the I²C Slave Read/Write flag. This flag determines whether the master device wishes to transmit or receive data from the I²C bus. When the transmitted address and slave address is match, that is when the HAAS flag is set high, the slave device will check the SRW flag to determine whether it should be in transmit mode or receive mode. If the SRW flag is high, the master is requesting to read data from the bus, so the slave device should be in transmit mode. When the SRW flag is zero, the master will write data to the bus, therefore the slave device should be in receive mode to read this data.

Bit 1 **IAMWU:** I²C Address Match Wake-up control
0: Disable
1: Enable – must be cleared by the application program after wake-up
This bit should be set to 1 to enable the I²C address match wake-up from the SLEEP or IDLE Mode. If the IAMWU bit has been set before entering either the SLEEP or IDLE mode to enable the I²C address match wake-up, then this bit must be cleared by the application program after wake-up to ensure correction device operation.

Bit 0 **RXAK:** I²C bus receive acknowledge flag
0: Slave receives acknowledge flag
1: Slave does not receive acknowledge flag
The RXAK flag is the receiver acknowledge flag. When the RXAK flag is "0", it means that a acknowledge signal has been received at the 9th clock, after 8 bits of data have been transmitted. When the slave device is in the transmit mode, the slave device checks the RXAK flag to determine if the master receiver wishes to receive the next byte. The slave transmitter will therefore continue sending out data until the RXAK flag is "1". When this occurs, the slave transmitter will release the SDA line to allow the master to send a STOP signal to release the I²C Bus.

I²C Bus Communication

Communication on the I²C bus requires four separate steps, a START signal, a slave device address transmission, a data transmission and finally a STOP signal. When a START signal is placed on the I²C bus, all devices on the bus will receive this signal and be notified of the imminent arrival of data on the bus. The first seven bits of the data will be the slave address with the first bit being the MSB. If the address of the slave device matches that of the transmitted address, the HAAS bit in the SIMC1 register will be set and an SIM interrupt will be generated. After entering the interrupt service routine, the slave device must first check the condition of the HAAS and SIMTOF bits to determine whether the interrupt source originates from either an address match or the completion of an 8-bit data transfer or the I²C bus time-out occurrence. During a data transfer, note that after the 7-bit slave address has been transmitted, the following bit, which is the 8th bit, is the read/write bit whose value will be placed in the SRW bit. This bit will be checked by the slave device to determine whether to go into transmit or receive mode. Before any transfer of data to or from the I²C bus, the microcontroller must initialise the bus; the following are steps to achieve this:

- Step 1

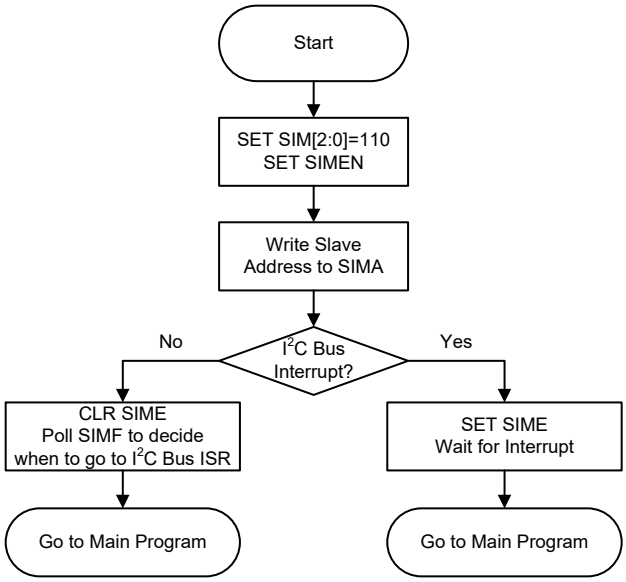
  Set the SIM2~SIM0 bits to "110" and SIMEN bit to "1" in the SIMC0 register to enable the I²C bus.

- Step 2

  Write the slave address of the device to the I²C bus address register SIMA.

- Step 3

  Set the SIME interrupt enable bit of the interrupt control register to enable the SIM interrupt.



I²C Bus Initialisation Flowchart

I²C Bus Start Signal

The START signal can only be generated by the master device connected to the I²C bus and not by the slave device. This START signal will be detected by all devices connected to the I²C bus. When detected, this indicates that the I²C bus is busy and therefore the HBB bit will be set. A START condition occurs when a high to low transition on the SDA line takes place when the SCL line remains high.

I²C Slave Address

The transmission of a START signal by the master will be detected by all devices on the I²C bus. To determine which slave device the master wishes to communicate with, the address of the slave device will be sent out immediately following the START signal. All slave devices, after receiving this 7-bit address data, will compare it with their own 7-bit slave address. If the address sent out by the master matches the internal address of the microcontroller slave device, then an internal SIM I²C bus interrupt signal will be generated. The next bit following the address, which is the 8th bit, defines the read/write status and will be saved to the SRW bit of the SIMC1 register. The slave device will then transmit an acknowledge bit, which is a low level, as the 9th bit. The slave device will also set the status flag HAAS when the addresses match.

As an SIM I²C bus interrupt can come from three sources, when the program enters the interrupt subroutine, the HAAS and SIMTOF bits should be examined to see whether the interrupt source has come from either a matching slave address, the completion of a data byte transfer or the I²C bus time-out occurrence. When a slave address is matched, the device must be placed in either the transmit mode and then write data to the SIMD register, or in the receive mode where it must implement a dummy read from the SIMD register to release the SCL line.

I²C Bus Read/Write Signal

The SRW bit in the SIMC1 register defines whether the master device wishes to read data from the I²C bus or write data to the I²C bus. The slave device should examine this bit to determine if it is to be a transmitter or a receiver. If the SRW flag is "1" then this indicates that the master device wishes to read data from the I²C bus, therefore the slave device must be setup to send data to the I²C bus as a transmitter. If the SRW flag is "0" then this indicates that the master wishes to send data to the I²C bus, therefore the slave device must be setup to read data from the I²C bus as a receiver.

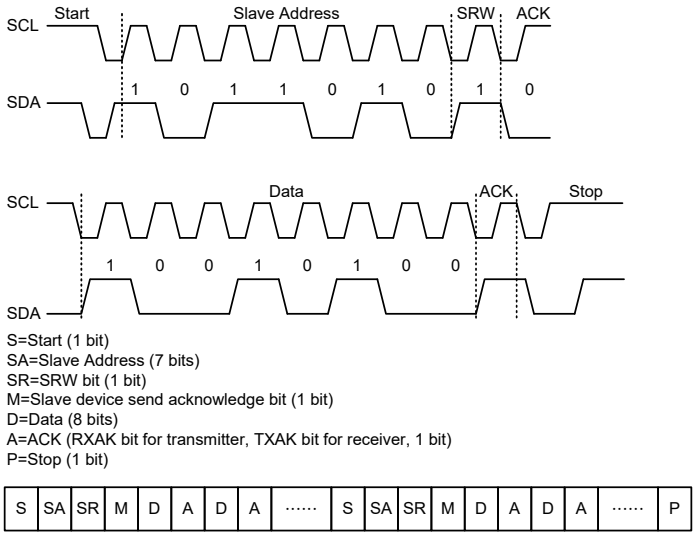
I²C Bus Slave Address Acknowledge Signal

After the master has transmitted a calling address, any slave device on the I²C bus, whose own internal address matches the calling address, must generate an acknowledge signal. The acknowledge signal will inform the master that a slave device has accepted its calling address. If no acknowledge signal is received by the master then a STOP signal must be transmitted by the master to end the communication. When the HAAS flag is high, the addresses have matched and the slave device must check the SRW flag to determine if it is to be a transmitter or a receiver. If the SRW flag is high, the slave device should be setup to be a transmitter so the HTX bit in the SIMC1 register should be set to "1". If the SRW flag is low, then the microcontroller slave device should be setup as a receiver and the HTX bit in the SIMC1 register should be cleared to "0".

I²C Bus Data and Acknowledge Signal

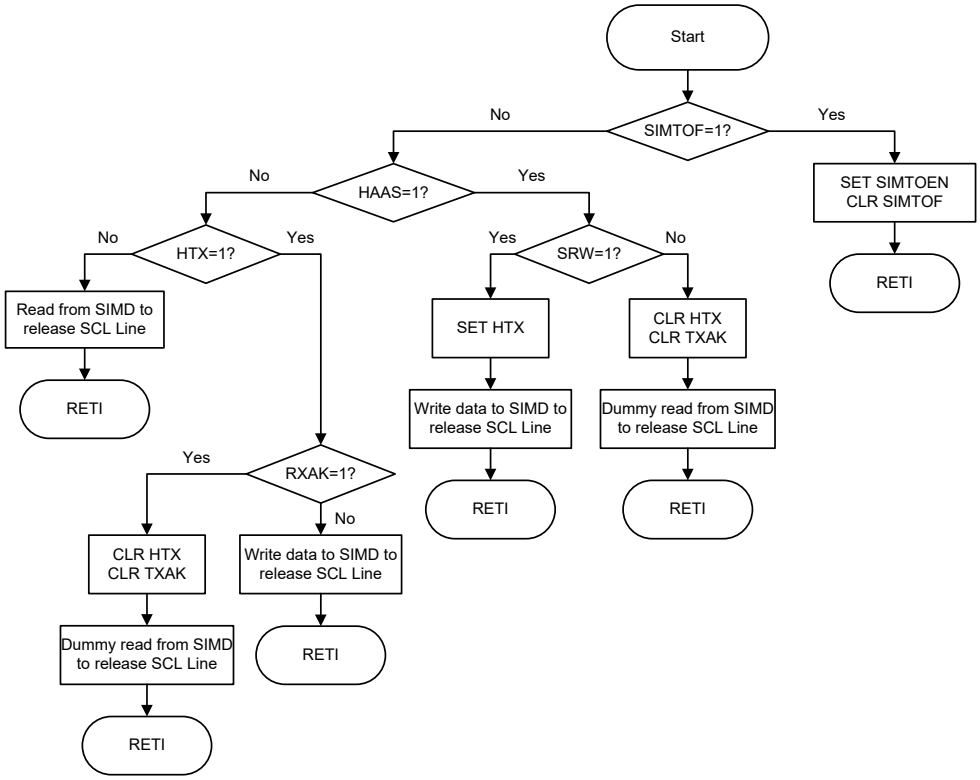
The transmitted data is 8-bit wide and is transmitted after the slave device has acknowledged receipt of its slave address. The order of serial bit transmission is the MSB first and the LSB last. After receipt of 8 bits of data, the receiver must transmit an acknowledge signal, level "0", before it can receive the next data byte. If the slave transmitter does not receive an acknowledge bit signal from the master receiver, then the slave transmitter will release the SDA line to allow the master to send a STOP signal to release the I²C Bus. The corresponding data will be stored in the SIMD register. If setup as a transmitter, the slave device must first write the data to be transmitted into the SIMD register. If setup as a receiver, the slave device must read the transmitted data from the SIMD register.

When the slave receiver receives the data byte, it must generate an acknowledge bit, known as TXAK, on the 9th clock. The slave device, which is setup as a transmitter will check the RXAK bit in the SIMC1 register to determine if it is to send another data byte, if not then it will release the SDA line and await the receipt of a STOP signal from the master.



Note: When a slave address is matched, the device must be placed in either the transmit mode and then write data to the SIMD register, or in the receive mode where it must implement a dummy read from the SIMD register to release the SCL line.

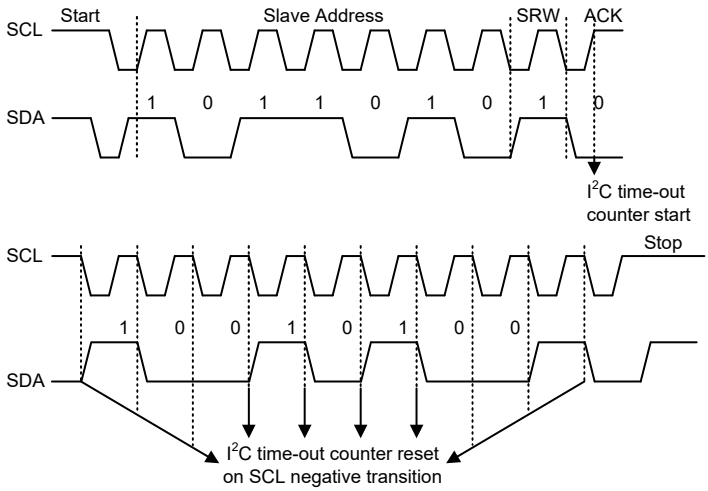
I²C Communication Timing Diagram



I²C Bus ISR Flowchart

I²C Time-out Control

In order to reduce the I²C lockup problem due to reception of erroneous clock sources, a time-out function is provided. If the clock source connected to the I²C bus is not received for a while, then the I²C circuitry and registers will be reset after a certain time-out period. The time-out counter starts to count on an I²C bus "START" & "address match" condition, and is cleared by an SCL falling edge. Before the next SCL falling edge arrives, if the time elapsed is greater than the time-out period specified by the SIMTOC register, then a time-out condition will occur. The time-out function will stop when an I²C "STOP" condition occurs.



I²C Time-out

When an I²C time-out counter overflow occurs, the counter will stop and the SIMTOEN bit will be cleared to zero and the SIMTOF bit will be set high to indicate that a time-out condition has occurred. The time-out condition will also generate an interrupt which uses the SIM interrupt vector. When an I²C time-out occurs, the I²C internal circuitry will be reset and the registers will be reset into the following condition:

Registers	After I²C Time-out
SIMD, SIMA, SIMC0	No change
SIMC1	Reset to POR condition

I²C Registers after Time-out

The SIMTOF flag can be cleared by the application program. There are 64 time-out period selections which can be selected using the SIMTOS5~SIMTOS0 bits in the SIMTOC register. The time-out duration is calculated by the formula: ((1~64)×(32/f_{SUB})). This gives a time-out period which ranges from about 1ms to 64ms.

- **SIMTOC Register**

Bit	7	6	5	4	3	2	1	0
Name	SIMTOEN	SIMTOF	SIMTOS5	SIMTOS4	SIMTOS3	SIMTOS2	SIMTOS1	SIMTOS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **SIMTOEN:** SIM I²C Time-out control
0: Disable
1: Enable

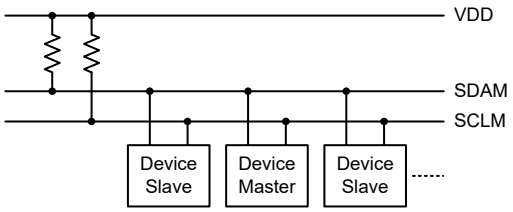
Bit 6 **SIMTOF:** SIM I²C Time-out flag
0: No time-out occurred
1: Time-out occurred

Bit 5~0 **SIMTOS5~SIMTOS0:** SIM I²C Time-out period selection
I²C Time-out clock source is f_{SUB}/32.
I²C Time-out period is equal to (SIMTOS[5:0]+1)×(32/f_{SUB}).

I²C Interface (Master Mode)

The device contains an independent I²C function. It is important not to confuse this independent I²C function with the additional one contained within the combined SIM function, which is described in another section of this datasheet. The main difference between them is that the independent I²C function operates in the master mode and the other one in the SIM operates in the slave mode.

The I²C interface is used to communicate with external peripheral devices such as sensors, EEPROM memory etc. Originally developed by Philips, it is a two line low speed serial interface for synchronous serial data transfer. The advantage of only two lines for communication, relatively simple communication protocol and the ability to accommodate multiple devices on the same bus has made it an extremely popular interface type for many applications.

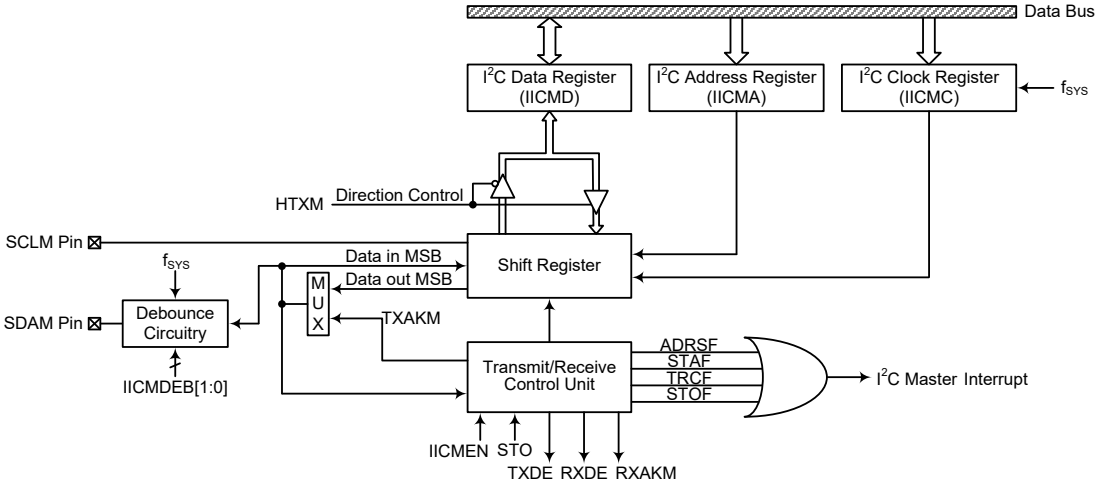


I²C Master/Slave Bus Connection

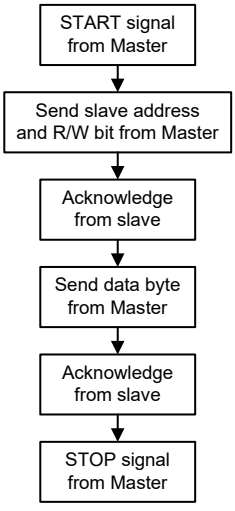
I²C Interface Operation

The I²C serial interface is a two line interface, a serial data line, SDAM, and serial clock line, SCLM. As many devices may be connected together on the same bus, their outputs are both open drain types. For this reason it is necessary that external pull-high resistors are connected to these outputs. Note that no chip select line exists, as each device on the I²C bus is identified by a unique address which will be transmitted and received on the I²C bus.

When two devices communicate with each other on the bidirectional I²C bus, one is known as the master device and one as the slave device. Both master and slave can transmit and receive data. However, it is the master device that has overall control of the bus. For the device, which only operates in master mode, there are two methods of transferring data on the I²C bus, the master transmit mode and the master receive mode. The pull-high control function pin-shared with SCLM/SDAM pin is still applicable even if the I²C device is activated and the related internal pull-high function could be controlled by its corresponding pull-high control register. It is suggested that the device should not enter the IDLE/SLEEP mode during the I²C communication.



I²C Block Diagram



I²C Interface Operation

I²C Registers

There are four control registers associated with the I²C bus, IICMC0, IICMC1, IICMC2 and IICMC, one address register IICMA and one data register, IICMD.

Register Name	Bit							
	7	6	5	4	3	2	1	0
IICMC0	—	—	—	—	IICMDEB1	IICMDEB0	IICMEN	—
IICMC1	SCLI	SDAI	STO	HTXM	TXAKM	RXDE	TXDE	RXAKM
IICMC2	TRCF	STOF	STAF	ADRSF	TRCE	STOE	STAE	ADRSE
IICMD	D7	D6	D5	D4	D3	D2	D1	D0
IICMA	IICMA6	IICMA5	IICMA4	IICMA3	IICMA2	IICMA1	IICMA0	—
IICMC	D7	D6	D5	D4	D3	D2	D1	D0

I²C Register List

I²C Data Register

The IICMD register is used to store the data being transmitted and received. Before the device writes data to the I²C bus, the actual data to be transmitted must be placed in the IICMD register. After the data is received from the I²C bus, the device can read it from the IICMD register. Any transmission or reception of data from the I²C bus must be made via the IICMD register.

- **IICMD Register**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	x	x	x	x	x	x	x	x

"x": Unknown

Bit 7~0 **D7~D0:** I²C serial bus data register bit 7 ~ bit 0

I²C Address Register

The IICMA register is used to specify the address of the target slave which the master wishes to communicate with.

- **IICMA Register**

Bit	7	6	5	4	3	2	1	0
Name	IICMA6	IICMA5	IICMA4	IICMA3	IICMA2	IICMA1	IICMA0	—
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	—
POR	0	0	0	0	0	0	0	—

Bit 7~1 **IICMA6~IICMA0:** I²C target slave address
IICMA6~IICMA0 is the I²C target slave address bit 6 ~ bit 0.

Bit 0 Unimplemented, read as "0"

I²C Control Registers

There are four control registers for the I²C interface, IICMC0, IICMC1, IICMC2 and IICMC. The IICMC0 register is used to control the enable/disable function and select the debounce time. The IICMC1 and IICMC2 registers contain the relevant flags which are used to indicate the I²C communication status. The IICMC register is used to control the I²C clock frequency.

• **IICMC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	IICMDEB1	IICMDEB0	IICMEN	—
R/W	—	—	—	—	R/W	R/W	R/W	—
POR	—	—	—	—	0	0	0	—

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **IICMDEB1~IICMDEB0:** I²C debounce time selection
00: No debounce
01: 2 system clock debounce
1x: 3 system clock debounce

Bit 1 **IICMEN:** I²C interface enable control
0: Disable
1: Enable
The bit is the overall on/off control for the I²C interface. When the IICMEN bit is cleared to zero to disable the I²C interface, the SDAM and SCLM lines will lose their I²C function and the I²C operating current will be reduced to a minimum value.

Bit 0 Unimplemented, read as "0"

• **IICMC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	SCLI	SDAI	STO	HTXM	TXAKM	RXDE	TXDE	RXAKM
R/W	R	R	R/W	R/W	R/W	R	R	R
POR	x	x	0	0	0	0	1	0

"x": Unknown

Bit 7 **SCLI:** I²C clock status monitor
0: Low level status
1: High level status

Bit 6 **SDAI:** I²C data status monitor
0: Low level status
1: High level status

Bit 5 **STO:** STOP condition control
0: No action
1: Send a STOP condition
When the STO bit is set high, the master device will transmit a STOP condition to terminate data transmission.

Bit 4 **HTXM:** I²C master device transmitter/receiver selection
0: Master device is the receiver
1: Master device is the transmitter

Bit 3 **TXAKM:** I²C bus transmit acknowledge flag in receive mode
0: Master does not send acknowledge flag
1: Master sends acknowledge flag
The TXAKM flag is the transmit acknowledge flag in the receive mode. After the master device has received 8 bits of data, this flag will be transmitted to the bus on the 9th clock from the master device. The master device must always set the TXAKM bit to "1" before further data is received.

Bit 2 **RXDE:** I²C data register empty flag in receive mode
0: Data register IICMD is empty
1: Data register IICMD is not empty
In the master receive mode, data is sent from the slave device. If the master device has received a complete new data byte, the RXDE bit will be set to "1" and the SCLM line will remain at a logic low state. When this occurs, the data in the IICMD register should be read to automatically clear the RXDE flag and start the next data reception.

Bit 1 **TXDE:** I²C data register empty flag in transmit mode
0: Data register IICMD is empty
1: Data register IICMD is not empty
In the master transmit mode, when the TXDE bit is set to "1", it indicates that the IICMD register is empty which holds the SCLM line at a logic low state. When this occurs, data should be written to the IICMD register to automatically clear the TXDE flag and start the next data transmission.

Bit 0 **RXAKM:** I²C bus receive acknowledge flag in transmit mode
0: Master does not receive acknowledge flag
1: Master receives acknowledge flag
The RXAKM flag is the receiver acknowledge flag from the slave in the transmit mode. When the RXAKM flag is "1", it means that a acknowledge signal has been received at the 9th clock, after the master device has transmitted 8 bits of data. The master transmitter then determines if it continues writing further data or sends a STOP signal to terminate data transmission.

• IICMC2 Register

Bit	7	6	5	4	3	2	1	0
Name	TRCF	STOF	STAF	ADRSF	TRCE	STOE	STAE	ADRSE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **TRCF:** I²C bus data transfer/receive completion flag
0: No completion of a 9-bit data transfer/receive
1: Completion of a 9-bit data transfer/receive
The TRCF flag is the data transfer/receive completion flag. After completion of a 9-bit data transfer/receive the flag will go high and an interrupt will be generated.

Bit 6 **STOF:** STOP condition flag
0: STOP condition no completion
1: STOP condition completion

Bit 5 **STAF:** START condition flag
0: START condition no completion
1: START condition completion

Bit 4 **ADRSF:** Address transmit flag
0: Address frame has not been transmitted
1: Address frame has been transmitted
This bit will be set high after the master receives the address frame acknowledgement bit sent from the slave.

Bit 3 **TRCE:** I²C bus data transfer/receive completion interrupt control
0: Disable
1: Enable

Bit 2 **STOE:** STOP condition interrupt control
0: Disable
1: Enable

Bit 1 **STAE:** START condition interrupt control
0: Disable
1: Enable

Bit 0 **ADRSE:** Address transmit interrupt control
0: Disable
1: Enable

• **IICMC Register**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0:** I²C Clock register bit 7 ~ bit 0

I²C Clock=f_{SYS}/((IICMC[7:0]+1)×2)

Note: The IICMC[7:0] settings depend upon the debounce time settings.

If IICMDEB[1:0]=00, then the IICMC[7:0] should be greater than 0;

If IICMDEB[1:0]=01, then the IICMC[7:0] should be greater than 2;

If IICMDEB[1:0]=10 or 11, then the IICMC[7:0] should be greater than 3.

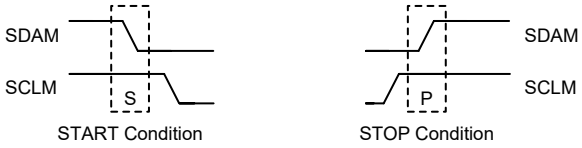
I²C START and STOP Conditions

A master device can initialize a transfer by sending a START signal and terminate the transfer with a STOP signal.

A START signal is usually referred to as the "S" bit, which is defined as a High to Low transition on the SDAM line while the SCLM line is high.

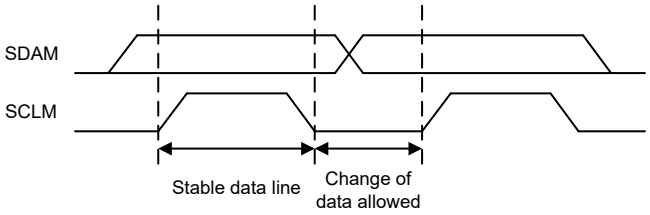
A STOP signal is usually referred to as the "P" bit, which is defined as a Low to High transition on the SDAM line while SCLM is high.

A repeated START signal allows the I²C interface to communicate with another slave device or with the same device but in a different transfer direction without releasing the I²C bus control.



I²C Data Validity

The data on the SDAM line must be stable during the high period of the SCLM clock. The SDAM data state can only be changed when the clock signal on the SCLM line is in a logic low state.

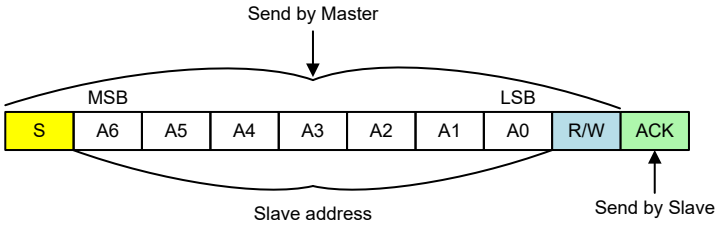


I²C Address Format

The address format is composed of the 7-bit length slave address, which the master device wants to communicate with, an R/W bit, and an ACK bit. The R/W bit defines the direction of the data transfer.

- R/W=0 (write): The master transmits data to the addressed slave.
- R/W=1 (read): The master receives data from the addressed slave.

The slave address can be assigned using the IICMA register. The slave device sends back the acknowledge bit (ACK) if its slave address matches the transmitted address sent by the master.

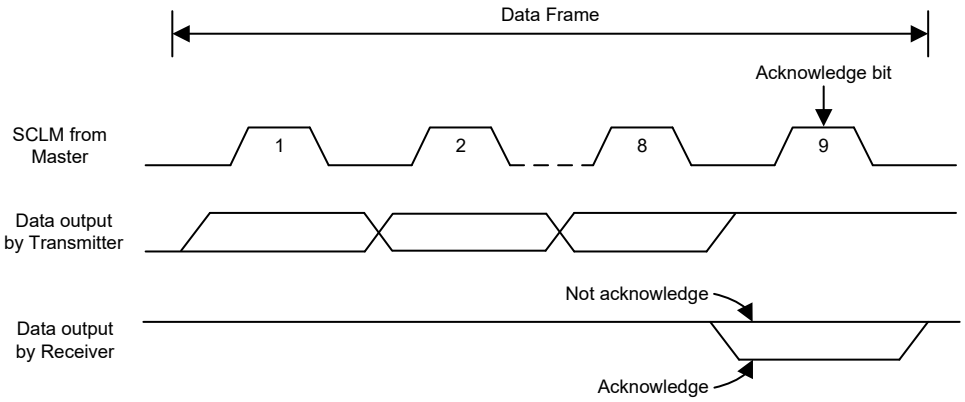


I²C Data Transfer and Acknowledge

Once the slave device address has been matched, the data can be transmitted to or received from the slave device according to the transfer direction specified by the R/W bit. Each byte is followed by an acknowledge bit on the 9th SCLM clock.

If the slave device returns a not-acknowledge (NACK) signal to the master device, the master device can generate a STOP signal to terminate the data transfer or generate a repeated START signal to restart the transfer.

If the master device sends a not-acknowledge (NACK) signal to the slave device, the slave device should release the SDAM line for the master device to generate a STOP signal to terminate the transfer.



I²C Transmit Mode

Start Condition

Users write the target slave address into the IICMA register after setting the IICMEN bit in the IICMC1 register. When a START condition occurs, the address frame is sent out.

Address Frame

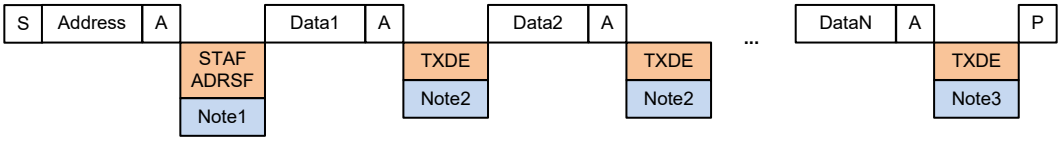
The ADRSF bit in the IICMC2 register will be set high after the address frame is sent by the master and the acknowledge signal from the address-matched slave is received. In order to send the subsequent data frame, the ADRSF flag must be cleared to zero if it has been set to 1.

Data Frame

The TXDE bit in the IICMC1 register is set to 1 to indicate that the IICMD register is empty, which results in the SCLM line remained at a logic low state. When this occurs, new data should then be written to the IICMD register to automatically clear the TXDE flag and start the next data transfer.

Stop/Continue Transmitting

After the last data byte has been transmitted, the STO bit in the IICMC1 register can be set high to terminate the transmission, or re-assign another slave device by configuring the IICMA register to restart a new transmission.



Note1: Cleared by software.
Note2: Cleared by writing the IICMD register.
Note3: Cleared automatically by hardware by sending a STOP condition.

I²C Receive Mode

Start Condition

Users write the target slave address into the IICMA register after setting the IICMEN bit in the IICMC1 register. When a START condition occurs, the address frame is sent out.

Address Frame

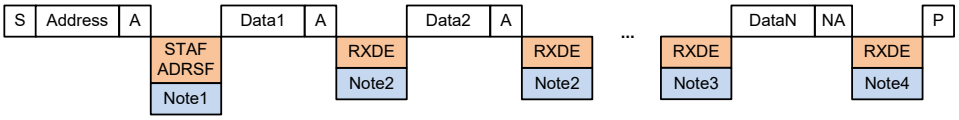
The ADRSF bit in the IICMC2 register will be set high after the address frame is sent by the master and the acknowledge signal from the address-matched slave is received. In order to receive the subsequent data frame from the slave, the ADRSF flag must be cleared to zero if it has been set to 1.

Data Frame

In the master receive mode, data is transmitted from the slave device. If the device receives a complete new data byte, the RXDE bit in the IICMC1 register will be set to 1 and the SCLM line will remain at a logic low state. When this occurs, data from the IICMD register should be read to automatically clear the RXDE flag and start the next data reception.

Stop/Continue Receiving

Before receiving the last data byte, the master should send an NACK signal to the slave by clearing the TXAKM bit in the IICMC1 register to 0. After the last data byte has been received from the slave, the master will hold the SCLM line at a logic low state following after an NACK signal sent by the master device to the slave. The STOP bit in the IICMC1 register can be set to 1 to terminate the data transmission.



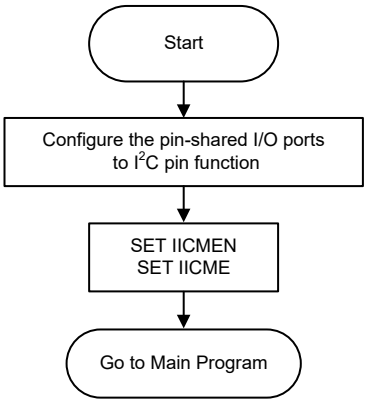
Note1: Cleared by software.
Note2: Cleared by reading the IICMD register.
Note3: Cleared by reading the IICMD register. Set TXAKM=0 to send a NACK signal.
Note4: Cleared by reading the IICMD register. Set STO=1 to send a STOP signal.

I²C Communication Flow

Communication on the I²C bus requires the several separate procedures, Procedure 1 ~ Procedure 4.

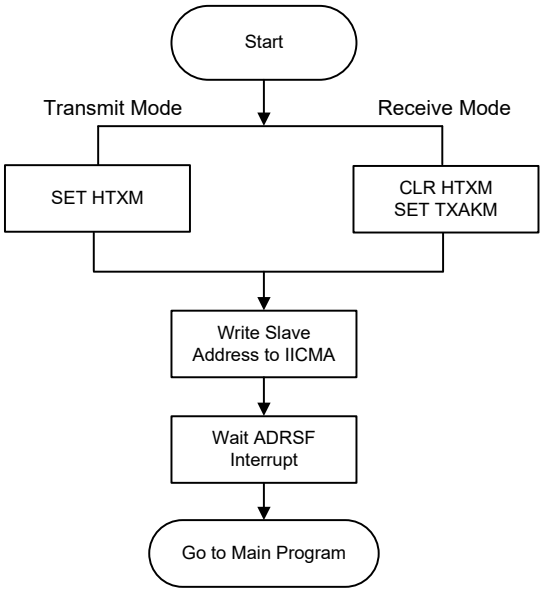
Procedure 1

1. Configure the corresponding pin-shared function as the I²C functional pins.
2. Set the IICMEN bit to enable the I²C function.
3. Set the IICME bit to enable the I²C master mode interrupt.



Procedure 2

1. Set the HTXM bit to select whether the master is in the Transmit mode or the Receive mode.
2. Write the target slave device address to the IICMA register.
3. When a write operation to the IICMA register is executed, a START signal will be generated and then the address will be sent out.
4. Wait until the ADRSF interrupt is triggered. If the ADRSF is set to 1, it indicates that the address has been successfully transmitted and then the slave sends an ACK signal to the acknowledge bit.

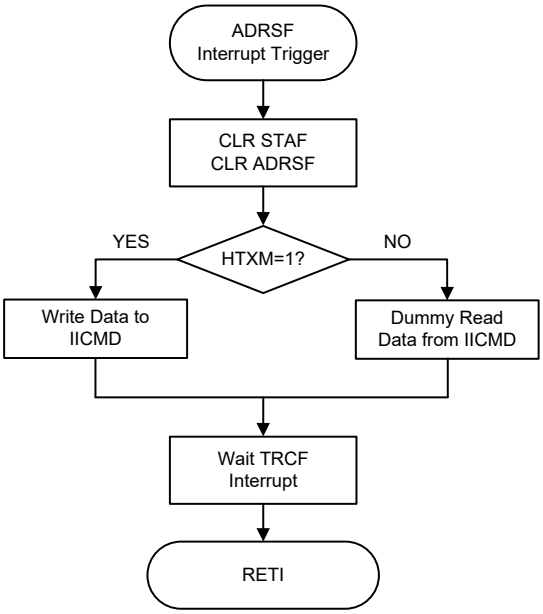


Procedure 3

1. After the ADRSF interrupt occurs, the ADRSF flag should be cleared.
2. Check the HTXM flag to determine whether the master is in the Transmit or Receive mode.
   - If in the Transmit mode:

     Write data to the IICMD register, then the master starts to send data to the slave. Wait until the TRCF interrupt is triggered, which indicates that data transmission is completed.
   - If in the Receive mode:

     Read data from the IICMD register, then the master starts to receive data from the slave. Wait until the TRCF interrupt is triggered, which indicates that data reception is completed.



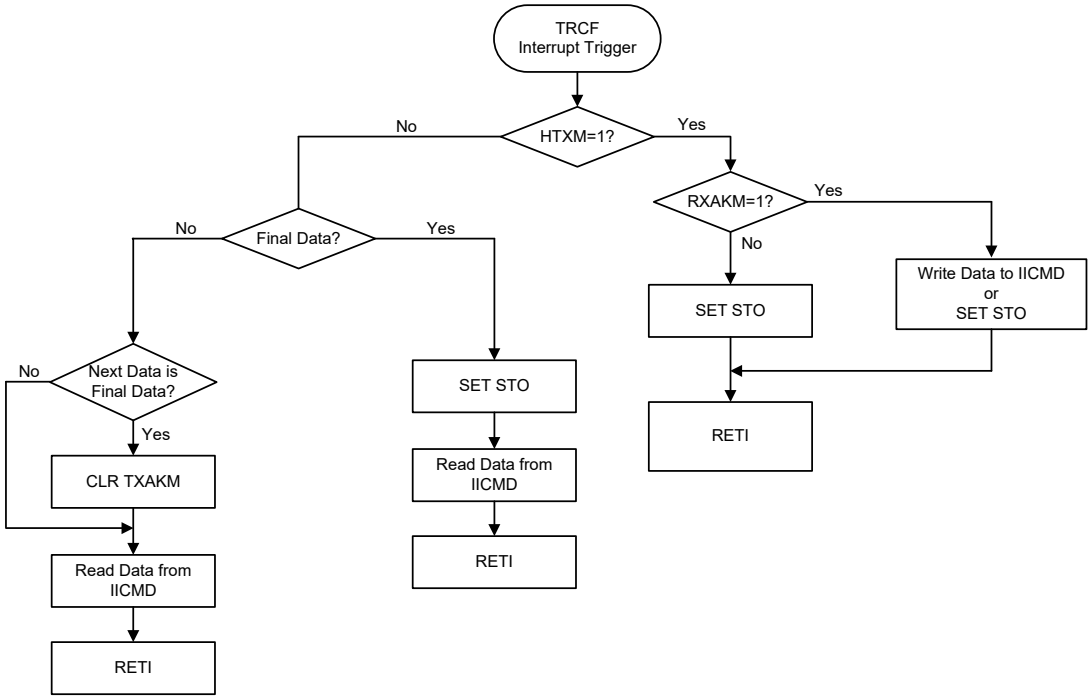
Procedure 4

1. When the TRCF interrupt is triggered, this indicates that data transmission or reception is completed.
2. Determine whether the master is in the Transmit or Receive mode by software.
   - If in the Transmit mode:

     Check the RXAKM flag to determine whether the slave has received data successfully.
     - When RXAKM=0, which indicates that the slave fails to receive data, the master will send a STOP signal by hardware.
     - When RXAKM=1, which indicates that the slave has received data successfully, if the master wishes to continue writing data, write data to the IICMD register again to send the next data. If the master wishes to terminate the transmission, set the STO to 1 by software to send a STOP signal.
   - If in the Receive mode:

     Check whether the next data is the final one.
     - If it is the final data, set the STO to 1 by software to send a STOP signal to terminate the reception and then read the IICMD register. This operation implements reading for the final data and does not send the timing sequence for receiving the next data.

- If it is not the final data, determine whether the next received data is the final one.
  - If no, read the previous data from the IICMD register and then send a timing sequence again to read the next data.
  - If yes, clear the TXAKM to 0 and then read the IICMD register. In the next reading process, the master will send an NACK signal to the acknowledge bit to indicate that it is the final received data because the TXAKM is equal to 0. At the end of the reception, the master will send a STOP signal by hardware.



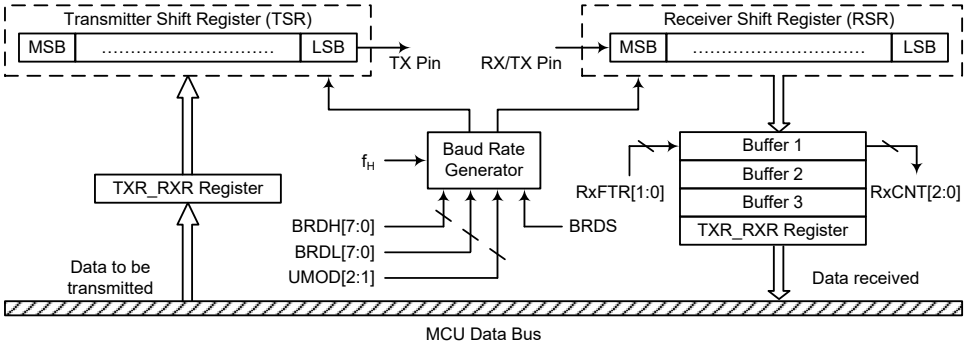
UART Interface

The device contains an integrated full-duplex or half-duplex asynchronous serial communications UART interface that enables communication with external devices that contain a serial interface. The UART function has many features and can transmit and receive data serially by transferring a frame of data with eight or nine data bits per transmission as well as being able to detect errors when the data is overwritten or incorrectly framed. The UART function possesses its own internal interrupt which can be used to indicate when a reception occurs or when a transmission terminates.

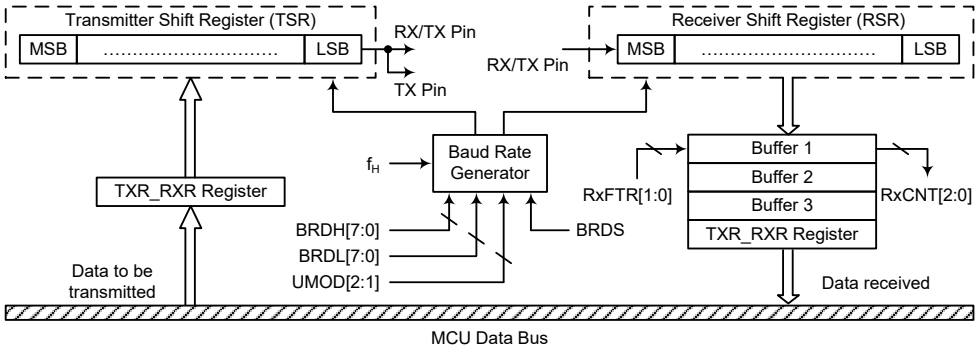
The integrated UART function contains the following features:

- Full-duplex or half-duplex (single wire mode), asynchronous communication
- 8 or 9 bits character length
- Even, odd, mark, space or no parity options
- One or two stop bits configurable for receiver
- Two stop bits for transmitter
- Baud rate generator with 16-bit prescaler
- Parity, framing, noise and overrun error detection
- Support for interrupt on address detect (last character bit=1)

- Separately enabled transmitter and receiver
- 4-byte Deep FIFO Receive Data Buffer
- 1-byte Deep FIFO Transmit Data Buffer
- RX/TX pin wake-up function
- Transmit and receive interrupts
- Interrupts can be triggered by the following conditions:
  - Transmitter Empty
  - Transmitter Idle
  - Receiver reaching FIFO trigger level
  - Receiver Overrun
  - Address Mode Detect



UART Data Transfer Block Diagram – SWM=0



UART Data Transfer Block Diagram – SWM=1

UART External Pins

To communicate with an external serial interface, the internal UART has two external pins known as TX and RX/TX, which are pin-shared with I/O or other pin functions. The TX and RX/TX pin function should first be selected by the corresponding pin-shared function selection register before the UART function is used. Along with the UARTEN bit, the TXEN and RXEN bits, if set, will configure these pins to transmitter output and receiver input conditions. At this time the internal pull-high resistor related to the transmitter output pin will be disabled, while the internal pull-high resistor related to the receiver input pin is controlled by the corresponding I/O pull-high function control bit. When the TX or RX/TX pin function is disabled by clearing the UARTEN, TXEN or RXEN bit, the TX or RX/TX pin will be set to a floating state. At this time whether the internal pull-high resistor is connected to the TX or RX/TX pin or not is determined by the corresponding I/O pull-high function control bit.

UART Single Wire Mode

The UART function also supports a Single Wire Mode communication which is selected using the SWM bit in the UCR3 register. When the SWM bit is set high, the UART function will be in the single wire mode. In the single wire mode, a single RX/TX pin can be used to transmit and receive data depending upon the corresponding control bits. When the RXEN bit is set high, the RX/TX pin is used as a receiver pin. When the RXEN bit is cleared to zero and the TXEN bit is set high, the RX/TX pin will act as a transmitter pin.

It is recommended not to set both the RXEN and TXEN bits high in the single wire mode. If both the RXEN and TXEN bits are set high, the RXEN bit will have the priority and the UART will act as a receiver.

It is important to note that the functional description in this UART chapter, which is described from the full-duplex communication standpoint, also applies to the half-duplex (single wire mode) communication except the pin usage. In the single wire mode, the TX pin mentioned in this chapter should be replaced by the RX/TX pin to understand the whole UART single wire mode function.

In the single wire mode, the data can also be transmitted on the TX pin in a transmission operation with proper software configurations. Therefore, the data will be output on the RX/TX and TX pins.

UART Data Transfer Scheme

The UART Data Transfer Block Diagram shows the overall data transfer structure arrangement for the UART. The actual data to be transmitted from the MCU is first transferred to the TXR_RXR register by the application program. The data will then be transferred to the Transmit Shift Register from where it will be shifted out, LSB first, onto the TX pin at a rate controlled by the Baud Rate Generator. Only the TXR_RXR register is mapped onto the MCU Data Memory, the Transmit Shift Register is not mapped and is therefore inaccessible to the application program.

Data to be received by the UART is accepted on the external RX/TX pin, from where it is shifted in, LSB first, to the Receiver Shift Register at a rate controlled by the Baud Rate Generator. When the shift register is full, the data will then be transferred from the shift register to the internal TXR_RXR register, where it is buffered and can be manipulated by the application program. Only the TXR_RXR register is mapped onto the MCU Data Memory, the Receiver Shift Register is not mapped and is therefore inaccessible to the application program.

It should be noted that the actual register for data transmission and reception only exists as a single shared register, TXR_RXR, in the Data Memory.

UART Status and Control Registers

There are nine control registers associated with the UART function. The SWM bit in the UCR3 register is used to enable/disable the UART Single Wire Mode. The USR, UCR1, UCR2, UFCR and RxCNT registers control the overall function of the UART, while the BRDH and BRDL registers control the Baud rate. The actual data to be transmitted and received on the serial interface is managed through the TXR_RXR data register.

Register Name	Bit							
	7	6	5	4	3	2	1	0
USR	PERR	NF	FERR	OERR	RIDLE	RXIF	TIDLE	TXIF
UCR1	UARTEN	BNO	PREN	PRT1	PRT0	TXBRK	RX8	TX8
UCR2	TXEN	RXEN	STOPS	ADDEN	WAKE	RIE	TIIE	TEIE
UCR3	—	—	—	—	—	—	—	SWM
TXR_RXR	TXRX7	TXRX6	TXRX5	TXRX4	TXRX3	TXRX2	TXRX1	TXRX0
BRDH	D7	D6	D5	D4	D3	D2	D1	D0

Register Name	Bit							
	7	6	5	4	3	2	1	0
BRDL	D7	D6	D5	D4	D3	D2	D1	D0
UFCR	—	—	UMOD2	UMOD1	UMOD0	BRDS	RxFTR1	RxFTR0
RxCNT	—	—	—	—	—	D2	D1	D0

UART Register List

- **USR Register**

The USR register is the status register for the UART, which can be read by the program to determine the present status of the UART. All flags within the USR register are read only. Further explanation on each of the flags is given below:

Bit	7	6	5	4	3	2	1	0
Name	PERR	NF	FERR	OERR	RIDLE	RXIF	TIDLE	TXIF
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	1	0	1	1

Bit 7 **PERR:** Parity error flag
0: No parity error is detected
1: Parity error is detected
The PERR flag is the parity error flag. When this read only flag is "0", it indicates a parity error has not been detected. When the flag is "1", it indicates that the parity of the received word is incorrect. This error flag is applicable only if the parity is enabled and the parity type (odd, even, mark or space) is selected. The flag can also be cleared by a software sequence which involves a read to the status register USR followed by an access to the TXR_RXR data register.

Bit 6 **NF:** Noise flag
0: No noise is detected
1: Noise is detected
The NF flag is the noise flag. When this read only flag is "0", it indicates no noise condition. When the flag is "1", it indicates that the UART has detected noise on the receiver input. The NF flag is set during the same cycle as the RXIF flag but will not be set in the case of as overrun. The NF flag can be cleared by a software sequence which will involve a read to the status register USR followed by an access to the TXR_RXR data register.

Bit 5 **FERR:** Framing error flag
0: No framing error is detected
1: Framing error is detected
The FERR flag is the framing error flag. When this read only flag is "0", it indicates that there is no framing error. When the flag is "1", it indicates that a framing error has been detected for the current character. The flag can also be cleared by a software sequence which will involve a read to the status register USR followed by an access to the TXR_RXR data register.

Bit 4 **OERR:** Overrun error flag
0: No overrun error is detected
1: Overrun error is detected
The OERR flag is the overrun error flag which indicates when the receiver buffer has overflowed. When this read only flag is "0", it indicates that there is no overrun error. When the flag is "1", it indicates that an overrun error occurs which will inhibit further transfers to the TXR_RXR receive data register. The flag is cleared by a software sequence, which is a read to the status register USR followed by an access to the TXR_RXR data register.

Bit 3 **RIDLE:** Receiver status
0: Data reception is in progress (Data being received)
1: No data reception is in progress (Receiver is idle)
The RIDLE flag is the receiver status flag. When this read only flag is "0", it indicates that the receiver is between the initial detection of the start bit and the completion of the stop bit. When the flag is "1", it indicates that the receiver is idle. Between the completion of the stop bit and the detection of the next start bit, the RIDLE bit is "1" indicating that the UART receiver is idle and the RX/TX pin stays in logic high condition.

Bit 2 **RXIF:** Receive TXR_RXR data register status
0: TXR_RXR data register is empty
1: TXR_RXR data register has available data and Receiver FIFO trigger level is reached
The RXIF flag is the receive data register status flag. When this read only flag is "0", it indicates that the TXR_RXR read data register is empty. When the flag is "1", it indicates that the TXR_RXR read data register contains new data and Receiver FIFO trigger level is reached. When the contents of the shift register are transferred to the TXR_RXR register and Receiver FIFO trigger level is reached, an interrupt is generated if RIE=1 in the UCR2 register. If one or more errors are detected in the received word, the appropriate receive-related flags NF, FERR, and/or PERR are set within the same clock cycle. The RXIF flag is cleared when the USR register is read with RXIF set, followed by a read from the TXR_RXR register, and if the TXR_RXR register has no data available.

Bit 1 **TIDLE:** Transmission idle
0: Data transmission is in progress (Data being transmitted)
1: No data transmission is in progress (Transmitter is idle)
The TIDLE flag is known as the transmission complete flag. When this read only flag is "0", it indicates that a transmission is in progress. This flag will be set high when the TXIF flag is "1" and when there is no transmit data or break character being transmitted. When TIDLE is equal to "1", the TX pin becomes idle with the pin state in logic high condition. The TIDLE flag is cleared by reading the USR register with TIDLE set and then writing to the TXR_RXR register. The flag is not generated when a data character or a break is queued and ready to be sent.

Bit 0 **TXIF:** Transmit TXR_RXR data register status
0: Character is not transferred to the transmit shift register
1: Character has transferred to the transmit shift register (TXR_RXR data register is empty)
The TXIF flag is the transmit data register empty flag. When this read only flag is "0", it indicates that the character is not transferred to the transmitter shift register. When the flag is "1", it indicates that the transmitter shift register has received a character from the TXR_RXR data register. The TXIF flag is cleared by reading the UART status register (USR) with TXIF set and then writing to the TXR_RXR data register. Note that when the TXEN bit is set, the TXIF flag will also be set since the transmit data register is not yet full.

• UCR1 Register

The UCR1 register together with the UCR2 and UCR3 register are the three UART control registers that are used to set the various options for the UART function, such as overall on/off control, parity control, data transfer bit length, single wire mode communication etc. Further explanation on each of the bits is given below:

Bit	7	6	5	4	3	2	1	0
Name	UARTEN	BNO	PREN	PRT1	PRT0	TXBRK	RX8	TX8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	W
POR	0	0	0	0	0	0	x	0

"x": unknown

Bit 7 **UARTEN:** UART function enable control
0: Disable UART. TX and RX/TX pins are in a floating state
1: Enable UART. TX and RX/TX pins function as UART pins

The UARTEN bit is the UART enable bit. When this bit is equal to "0", the UART will be disabled and the RX/TX pin as well as the TX pin will be set in a floating state. When the bit is equal to "1", the UART will be enabled and the TX and RX/TX pins will function as defined by the SWM mode selection bit together with the TXEN and RXEN enable control bits.

When the UART is disabled, it will empty the buffer so any character remaining in the buffer will be discarded. In addition, the value of the baud rate counter will be reset. If the UART is disabled, all error and status flags will be reset. Also the TXEN, RXEN, TXBRK, RXIF, OERR, FERR, PERR and NF bits as well as the RxCNT register will be cleared, while the TIDLE, TXIF and RIDLE bits will be set. Other control bits in UCR1, UCR2, UCR3, UFCR, BRDH and BRDL registers will remain unaffected. If the UART is active and the UARTEN bit is cleared, all pending transmissions and receptions will be terminated and the module will be reset as defined above. When the UART is re-enabled, it will restart in the same configuration.

Bit 6 **BNO:** Number of data transfer bits selection
0: 8-bit data transfer
1: 9-bit data transfer

This bit is used to select the data length format, which can have a choice of either 8-bit or 9-bit format. When this bit is equal to "1", a 9-bit data length format will be selected. If the bit is equal to "0", then an 8-bit data length format will be selected. If 9-bit data length format is selected, then bits RX8 and TX8 will be used to store the 9th bit of the received and transmitted data respectively.

Note that the 9th bit of data if BNO=1, or the 8th bit of data if BNO=0, which is used as the parity bit, does not transfer to RX8 or TXRX7 respectively when the parity function is enabled.

Bit 5 **PREN:** Parity function enable control
0: Parity function is disabled
1: Parity function is enabled

This is the parity enable bit. When this bit is equal to "1", the parity function will be enabled. If the bit is equal to "0", then the parity function will be disabled. Replace the most significant bit position with a parity bit.

Bit 4~3 **PRT1~PRT0:** Parity type selection bits
00: Even parity for parity generator
01: Odd parity for parity generator
10: Mark parity for parity generator
11: Space parity for parity generator

These bits are the parity type selection bits. When these bits are equal to 00b, even parity type will be selected. If these bits are equal to 01b, then odd parity type will be selected. If these bits are equal to 10b, then a 1 (Mark) in the parity bit location will be selected. If these bits are equal to 11b, then a 0 (Space) in the parity bit location will be selected.

Bit 2 **TXBRK:** Transmit break character
0: No break character is transmitted
1: Break characters transmit

The TXBRK bit is the Transmit Break Character bit. When this bit is "0", there are no break characters and the TX pin operates normally. When the bit is "1", there are transmit break characters and the transmitter will send logic zeros. When this bit is equal to "1", after the buffered data has been transmitted, the transmitter output is held low for a minimum of a 13-bit length and until the TXBRK bit is reset.

Bit 1 **RX8:** Receive data bit 8 for 9-bit data transfer format (read only)

This bit is only used if 9-bit data transfers are used, in which case this bit location will store the 9th bit of the received data known as RX8. The BNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.

Bit 0 **TX8:** Transmit data bit 8 for 9-bit data transfer format (write only)
This bit is only used if 9-bit data transfers are used, in which case this bit location will store the 9th bit of the transmitted data known as TX8. The BNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.

• UCR2 Register

The UCR2 register is the second of the two UART control registers and serves several purposes. One of its main functions is to control the basic enable/disable operation of the UART Transmitter and Receiver as well as enabling the various UART interrupt sources. The register also serves to control the receiver STOP bit number selection, receiver wake-up enable and the address detect enable. Further explanation on each of the bits is given below.

Bit	7	6	5	4	3	2	1	0
Name	TXEN	RXEN	STOPS	ADDEN	WAKE	RIE	TIIE	TEIE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **TXEN:** UART Transmitter enabled control
0: UART transmitter is disabled
1: UART transmitter is enabled
The bit named TXEN is the Transmitter Enable Bit. When this bit is equal to "0", the transmitter will be disabled with any pending data transmissions being aborted. In addition the buffers will be reset. In this situation the TX pin will be set in a floating state.
If the TXEN bit is equal to "1" and the UARTEN bit is also equal to "1", the transmitter will be enabled and the TX pin will be controlled by the UART. Clearing the TXEN bit during a transmission will cause the data transmission to be aborted and will reset the transmitter. If this situation occurs, the TX pin will be set in a floating state.

Bit 6 **RXEN:** UART Receiver enabled control
0: UART receiver is disabled
1: UART receiver is enabled
The bit named RXEN is the Receiver Enable Bit. When this bit is equal to "0", the receiver will be disabled with any pending data receptions being aborted. In addition the receive buffers will be reset. In this situation the RX/TX pin will be set in a floating state. If the RXEN bit is equal to "1" and the UARTEN bit is also equal to "1", the receiver will be enabled and the RX/TX pin will be controlled by the UART. Clearing the RXEN bit during a reception will cause the data reception to be aborted and will reset the receiver. If this situation occurs, the RX/TX pin will be set in a floating state.

Bit 5 **STOPS:** Number of Stop bits selection for receiver
0: One stop bit format is used
1: Two stop bits format is used
This bit determines if one or two stop bits are to be used for receiver. When this bit is equal to "1", two stop bits are used. If this bit is equal to "0", then only one stop bit is used. Two stop bits are used for transmitter.

Bit 4 **ADDEN:** Address detect function enable control
0: Address detect function is disabled
1: Address detect function is enabled
The bit named ADDEN is the address detect function enable control bit. When this bit is equal to "1", the address detect function is enabled. When it occurs, if the 8th bit, which corresponds to TXRX7 if BNO=0 or the 9th bit, which corresponds to RX8 if BNO=1, has a value of "1", then the received word will be identified as an address, rather than data. If the corresponding interrupt is enabled, an interrupt request will be generated each time the received word has the address bit set, which is the 8th or 9th bit depending on the value of BNO. If the address bit known as the 8th or 9th bit of the received word is "0" with the address detect function being enabled, an interrupt will not be generated and the received data will be discarded.

Bit 3 **WAKE:** RX/TX pin wake-up UART function enable control
0: RX/TX pin wake-up UART function is disabled
1: RX/TX pin wake-up UART function is enabled
This bit is used to control the wake-up UART function when a falling edge on the RX/TX pin occurs. Note that this bit is only available when the UART clock (f_H) is switched off. There will be no RX/TX pin wake-up UART function if the UART clock (f_H) exists. If the WAKE bit is set to 1 as the UART clock (f_H) is switched off, a UART wake-up request will be initiated when a falling edge on the RX/TX pin occurs. When this request happens and the corresponding interrupt is enabled, an RX/TX pin wake-up UART interrupt will be generated to inform the MCU to wake up the UART function by switching on the UART clock (f_H) via the application program. Otherwise, the UART function cannot resume even if there is a falling edge on the RX/TX pin when the WAKE bit is cleared to 0.

Bit 2 **RIE:** Receiver interrupt enable control
0: Receiver related interrupt is disabled
1: Receiver related interrupt is enabled
This bit enables or disables the receiver interrupt. If this bit is equal to "1" and when the receiver overrun flag OERR or receive data available flag RXIF is set, the UART interrupt request flag will be set. If this bit is equal to "0", the UART interrupt request flag will not be influenced by the condition of the OERR or RXIF flags.

Bit 1 **TIIE:** Transmitter Idle interrupt enable control
0: Transmitter idle interrupt is disabled
1: Transmitter idle interrupt is enabled
This bit enables or disables the transmitter idle interrupt. If this bit is equal to "1" and when the transmitter idle flag TIDLE is set, due to a transmitter idle condition, the UART interrupt request flag will be set. If this bit is equal to "0", the UART interrupt request flag will not be influenced by the condition of the TIDLE flag.

Bit 0 **TEIE:** Transmitter Empty interrupt enable control
0: Transmitter empty interrupt is disabled
1: Transmitter empty interrupt is enabled
This bit enables or disables the transmitter empty interrupt. If this bit is equal to "1" and when the transmitter empty flag TXIF is set, due to a transmitter empty condition, the UART interrupt request flag will be set. If this bit is equal to "0", the UART interrupt request flag will not be influenced by the condition of the TXIF flag.

• UCR3 Register

The UCR3 register is used to enable the UART Single Wire Mode communication. As the name suggests in the single wire mode the UART communication can be implemented in one single line, RX/TX, together with the control of the RXEN and TXEN bits in the UCR2 register.

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	—	SWM
R/W	—	—	—	—	—	—	—	R/W
POR	—	—	—	—	—	—	—	0

Bit 7~1 Unimplemented, read as "0"

Bit 0 **SWM:** Single Wire Mode enable control
0: Disable, the RX/TX pin is used as UART receiver function only
1: Enable, the RX/TX pin can be used as UART receiver or transmitter function controlled by the RXEN and TXEN bits
Note that when the Single Wire Mode is enabled, if both the RXEN and TXEN bits are high, the RX/TX pin will just be used as UART receiver input.

• **TXR_RXR Register**

The TXR_RXR register is the data register which is used to store the data to be transmitted on the TX pin or being received from the RX/TX pin.

Bit	7	6	5	4	3	2	1	0
Name	TXRX7	TXRX6	TXRX5	TXRX4	TXRX3	TXRX2	TXRX1	TXRX0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	x	x	x	x	x	x	x	x

"x": unknown

Bit 7~0 **TXRX7~TXRX0:** UART Transmit/Receive Data bit 7 ~ bit 0

• **BRDH Register**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0:** Baud rate divider high byte

The baud rate divider BRD (BRDH/BRDL) defines the UART clock divider ratio.

Baud Rate=f_H/(BRD+UMOD/8)

BRD=16~65535 or 8~65535 depending on BRDS

Note: 1. The BRD value should not be set to less than 16 when BRDS=0 or less than 8 when BRDS=1, otherwise errors may occur.

2. The BRDL must be written first and then BRDH, otherwise errors may occur.

3. The BRDH register should not be modified during data transmission process.

• **BRDL Register**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0:** Baud rate divider low byte

The baud rate divider BRD (BRDH/BRDL) defines the UART clock divider ratio.

Baud Rate=f_H/(BRD+UMOD/8)

BRD=16~65535 or 8~65535 depending on BRDS

Note: 1. The BRD value should not be set to less than 16 when BRDS=0 or less than 8 when BRDS=1, otherwise errors may occur.

2. The BRDL must be written first and then BRDH, otherwise errors may occur.

3. The BRDL register should not be modified during data transmission process.

• **UFCR Register**

The UFCR register is the FIFO control register which is used for UART modulation control, BRD range selection and trigger level selection for RXIF and interrupt.

Bit	7	6	5	4	3	2	1	0
Name	—	—	UMOD2	UMOD1	UMOD0	BRDS	RxFTR1	RxFTR0
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5~3 **UMOD2~UMOD0:** UART Modulation Control bits

The modulation control bits are used to correct the baud rate of the received or transmitted UART signal. These bits determine if the extra UART clock cycle should

be added in a UART bit time. The UMOD2~UMOD0 will be added to internal accumulator for every UART bit time. Until a carry to bit 3, the corresponding UART bit time increases a UART clock cycle.

Bit 2 **BRDS:** BRD range selection
0: BRD range is from 16 to 65535
1: BRD range is from 8 to 65535

The BRDS is used to control the sampling point in a UART bit time. If the BRDS bit is cleared to zero, the sampling point will be BRD/2, BRD/2+1×f_H, and BRD/2+2×f_H in a UART bit time. If the BRDS bit is set high, the sampling point will be BRD/2-1×f_H, BRD/2, and BRD/2+2×f_H in a UART bit time.

Note that the BRDS bit should not be modified during data transmission process.

Bit 1~0 **RxFTR1~RxFTR0:** Receiver FIFO trigger level (bytes)
00: 4 bytes in Receiver FIFO
01: 1 or more bytes in Receiver FIFO
10: 2 or more bytes in Receiver FIFO
11: 3 or more bytes in Receiver FIFO

For the receiver these bits define the number of received data bytes in the Receiver FIFO that will trigger the RXIF bit being set high, an interrupt will also be generated if the RIE bit is enabled. To prevent OERR from being set high, the receiver FIFO trigger level can be set to 2 bytes, avoiding an overrun state that cannot be processed by the program in time when more than 4 data bytes are received. After the reset the Receiver FIFO is empty.

- **RxCNT Register**

The RxCNT register is the counter used to indicate the number of received data bytes in the Receiver FIFO which have not been read by the MCU. This register is read only.

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	D2	D1	D0
R/W	—	—	—	—	—	R	R	R
POR	—	—	—	—	—	0	0	0

Bit 7~3 Unimplemented, read as "0"

Bit 2~0 **D2~D0:** Receiver FIFO counter

The RxCNT register is the counter used to indicate the number of received data bytes in the Receiver FIFO which is not read by the MCU. When Receiver FIFO receives one byte data, the RxCNT will increase by one; when the MCU reads one byte data from the Receiver FIFO, the RxCNT will decrease by one. If there are 4 bytes of data in the Receiver FIFO, the 5th data will be saved in the shift register. If there is 6th data, the 6th data will be saved in the shift register. But the RxCNT remains the value of 4. The RxCNT will be cleared when reset occurs or UARTEN=1. This register is read only.

Baud Rate Generator

To setup the speed of the serial data communication, the UART function contains its own dedicated baud rate generator. The baud rate is controlled by its own internal free running 16-bit timer, the period of which is determined by two factors. The first of these is the value placed in the BRDH/BRDL register and the second is the UART modulation control bits UMOD2~UMOD0. To prevent accumulated error of the receiver baud rate frequency, it is recommended to use two stop bits for resynchronization after each byte is received. If a baud rate BR is required with UART clock f_H.

f_H/BR = Integer Part + Fractional Part

The integer part is loaded into BRD (BRDH/BRDL). The fractional part is multiplied by 8 and rounded, then loaded into the UMOD bit field below:

BRD = TRUNC(f_H/BR)

UMOD = ROUND[MOD(f_H/BR)×8]

Therefore, the actual baud rate is calculated as follows:

Baud rate = f_H/[BRD+(UMOD/8)]

Calculating the Baud Rate and Error Values

For a clock frequency of 4MHz, determine the BRDH/BRDL register value, the actual baud rate and the error value for a desired baud rate of 230400.

From the above formula, the BRD = TRUNC(f_H/BR) = TRUNC(17.36111) = 17

The UMOD = ROUND[MOD(f_H/BR)×8] = ROUND(0.36111×8) = ROUND(2.88888) = 3

The actual Baud Rate = f_H/[BRD+(UMOD/8)] = 230215.83

Therefore the error is equal to (230215.83-230400)/230400 = -0.08%

Modulation Control Example

To get the best-fitting bit sequence for UART modulation control bits UMOD2~UMOD0, the following algorithm can be used: Firstly, the fractional part of the theoretical division factor is multiplied by 8. Then the product will be rounded and UMOD2~UMOD0 bits will be filled with the rounded value. The UMOD2~UMOD0 will be added to internal accumulator for every UART bit time. Until a carry to bit 3, the corresponding UART bit time increases a UART clock cycle. The following is an example using the fraction 0.36111 previously calculated: UMOD[2:0] = ROUND(0.36111×8) = 011b.

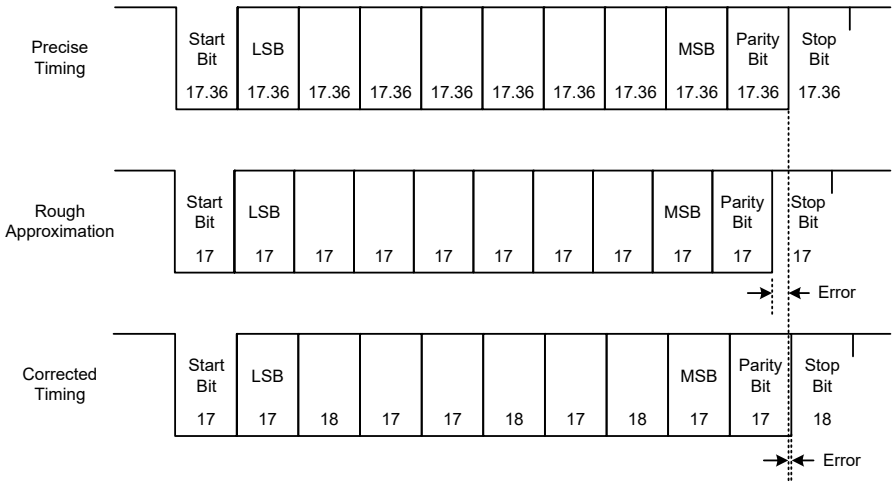
Fraction Addition	Carry to Bit 3	UART Bit Time Sequence	Extra UART Clock Cycle
0000b+0011b=0011b	No	Start bit	No
0011b+0011b=0110b	No	D0	No
0110b+0011b=1001b	Yes	D1	Yes
1001b+0011b=1100b	No	D2	No
1100b+0011b=1111b	No	D3	No
1111b+0011b=0010b	Yes	D4	Yes
0010b+0011b=0101b	No	D5	No
0101b+0011b=1000b	Yes	D6	Yes
1000b+0011b=1011b	No	D7	No
1011b+0011b=1110b	No	Parity bit	No
1110b+0011b=0001b	Yes	Stop bit	Yes

Baud Rate Correction Example

The following figure presents an example using a baud rate of 230400 generated with UART clock f_H. The data format for the following figure is: eight data bits, parity enabled, no address bit, two stop bits.

The following figure shows three different frames:

- The upper frame is the correct one, with a bit-length of 17.36 f_H cycles (4000000/230400=17.36).
- The middle frame uses a rough estimate, with 17 f_H cycles for the bit length.
- The lower frame shows a corrected frame using the best fit for the UART modulation control bits UMOD2~UMOD0.



UART Setup and Control

For data transfer, the UART function utilizes a non-return-to-zero, more commonly known as NRZ, format. This is composed of one start bit, eight or nine data bits, and one or two stop bits. Parity is supported by the UART hardware, and can be setup to be even, odd, mark, space or no parity. For the most common data format, 8 data bits along with no parity and one stop bit, denoted as 8, N, 1, is used as the default setting, which is the setting at power-on. The number of data bits along with the parity are setup by programming the BNO, PRT1~PRT0 and PREN bits. The transmitter always uses two stop bits while the receiver uses one or two stop bits which is determined by the STOPS bit. The baud rate used to transmit and receive data is setup using the internal 16-bit baud rate generator, while the data is transmitted and received LSB first. Although the UART transmitter and receiver are functionally independent, they both use the same data format and baud rate. In all cases stop bits will be used for data transmission.

Enabling/Disabling the UART Interface

The basic on/off function of the internal UART function is controlled using the UARTEN bit in the UCR1 register. If the UARTEN, TXEN and RXEN bits are set, then these two UART pins will act as normal TX output pin and RX/TX input pin respectively. If no data is being transmitted on the TX pin, then it will default to a logic high value.

Clearing the UARTEN bit will disable the TX and RX/TX pins and allow these two pins to be used as normal I/O or other pin-shared functional pins by configuring the corresponding pin-shared control bits. When the UART function is disabled the buffer will be reset to an empty condition, at the same time discarding any remaining residual data. Disabling the UART will also reset the error and status flags with bits TXEN, RXEN, TXBRK, RXIF, OERR, FERR, PERR and NF as well as register RxCNT being cleared while bits TIDLE, TXIF and RIDLE will be set. The remaining control bits in the UCR1, UCR2, UCR3, UFCR, BRDH and BRDL registers will remain unaffected. If the UARTEN bit in the UCR1 register is cleared while the UART is active, then all pending transmissions and receptions will be immediately suspended and the UART will be reset to a condition as defined above. If the UART is then subsequently re-enabled, it will restart again in the same configuration.

Data, Parity and Stop Bit Selection

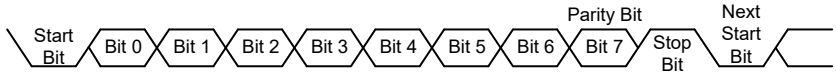
The format of the data to be transferred is composed of various factors such as data bit length, parity on/off, parity type, address bits and the number of stop bits. These factors are determined by the setup of various bits within the UCR1 and UCR2 registers. The BNO bit controls the number of data

bits which can be set to either 8 or 9, the PRT1~PRT0 bits control the choice of odd, even, mark or space parity, the PREN bit controls the parity on/off function and the STOPS bit decides whether one or two stop bits are to be used for the receiver, while the transmitter always uses two stop bits. The following table shows various formats for data transmission. The address bit, which is the MSB of the data byte, identifies the frame as an address character or data if the address detect function is enabled. The number of stop bits, which can be either one or two, is independent of the data length and is only configurable for the receiver. The transmitter uses two stop bits.

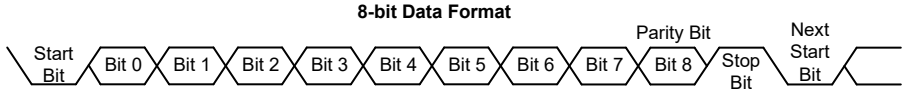
Start Bit	Data Bits	Address Bit	Parity Bit	Stop Bit
Example of 8-bit Data Formats				
1	8	0	0	1 or 2
1	7	0	1	1 or 2
1	7	1	0	1 or 2
Example of 9-bit Data Formats				
1	9	0	0	1 or 2
1	8	0	1	1 or 2
1	8	1	0	1 or 2

Transmitter Receiver Data Format

The following diagram shows the transmit and receive waveforms for both 8-bit and 9-bit data formats.



8-bit Data Format



9-bit Data Format

UART Transmitter

Data word lengths of either 8 or 9 bits can be selected by programming the BNO bit in the UCR1 register. When BNO bit is set, the word length will be set to 9 bits. In this case the 9th bit, which is the MSB, needs to be stored in the TX8 bit in the UCR1 register. At the transmitter core lies the Transmitter Shift Register, more commonly known as the TSR, whose data is obtained from the transmit data register, which is known as the TXR_RXR register. The data to be transmitted is loaded into this TXR_RXR register by the application program. The TSR register is not written to with new data until the stop bit from the previous transmission has been sent out. As soon as this stop bit has been transmitted, the TSR can then be loaded with new data from the TXR_RXR register, if it is available. It should be noted that the TSR register, unlike many other registers, is not directly mapped into the Data Memory area and as such is not available to the application program for direct read/write operations. An actual transmission of data will normally be enabled when the TXEN bit is set, but the data will not be transmitted until the TXR_RXR register has been loaded with data and the baud rate generator has defined a shift clock source. However, the transmission can also be initiated by first loading data into the TXR_RXR register, after which the TXEN bit can be set. When a transmission of data begins, the TSR is normally empty, in which case a transfer to the TXR_RXR register will result in an immediate transfer to the TSR. If during a transmission the TXEN bit is cleared, the transmission will immediately cease and the transmitter will be reset. The TX output pin can then be configured as the I/O or other pin-shared functions by configuring the corresponding pin-shared control bits.

Transmitting Data

When the UART is transmitting data, the data is shifted on the TX pin from the shift register, with the least significant bit first. In the transmit mode, the TXR_RXR register forms a buffer between the internal bus and the transmitter shift register. It should be noted that if 9-bit data format has been selected, then the MSB will be taken from the TX8 bit in the UCR1 register. The steps to initiate a data transfer can be summarized as follows:

- Make the correct selection of the BNO, PRT1~PRT0 and PREN bits to define the required word length and parity type. Two stop bits are used for the transmitter.
- Setup the BRDH and BRDL registers and the UMOD2~UMOD0 bits to select the desired baud rate.
- Set the TXEN bit to ensure that the TX pin is used as a UART transmitter pin.
- Access the USR register and write the data that is to be transmitted into the TXR_RXR register. Note that this step will clear the TXIF bit.

This sequence of events can now be repeated to send additional data.

It should be noted that when TXIF=0, data will be inhibited from being written to the TXR_RXR register. Clearing the TXIF flag is always achieved using the following software sequence:

1. A USR register access
2. A TXR_RXR register write execution

The read-only TXIF flag is set by the UART hardware and if set indicates that the TXR_RXR register is empty and that other data can now be written into the TXR_RXR register without overwriting the previous data. If the TEIE bit is set then the TXIF flag will generate an interrupt.

During a data transmission, a write instruction to the TXR_RXR register will place the data into the TXR_RXR register, which will be copied to the shift register at the end of the present transmission. When there is no data transmission in progress, a write instruction to the TXR_RXR register will place the data directly into the shift register, resulting in the commencement of data transmission, and the TXIF bit being immediately set. When a frame transmission is complete, which happens after stop bits are sent or after the break frame, the TIDLE bit will be set. To clear the TIDLE bit the following software sequence is used:

1. A USR register access
2. A TXR_RXR register write execution

Note that both the TXIF and TIDLE bits are cleared by the same software sequence.

Transmitting Break

If the TXBRK bit is set and the state keeps for a time greater than (BRD+1)×t_H while TIDLE=1, then break characters will be sent on the next transmission. Break character transmission consists of a start bit, followed by 13×N '0' bits and stop bits, where N=1, 2, etc. If a break character is to be transmitted then the TXBRK bit must be first set by the application program, and then cleared to generate the stop bits. Transmitting a break character will not generate a transmit interrupt. Note that a break condition length is at least 13 bits long. If the TXBRK bit is continually kept at a logic high level then the transmitter circuitry will transmit continuous break characters. After the application program has cleared the TXBRK bit, the transmitter will finish transmitting the last break character and subsequently send out two stop bits. The automatic logic highs at the end of the last break character will ensure that the start bit of the next frame is recognized.

UART Receiver

The UART is capable of receiving word lengths of either 8 or 9 bits. If the BNO bit is set, the word length will be set to 9 bits with the MSB being stored in the RX8 bit of the UCR1 register. At the receiver core lies the Receive Serial Shift Register, commonly known as the RSR. The data which is received on the RX/TX external input pin is sent to the data recovery block. The data recovery block operating speed is 16 times that of the baud rate, while the main receive serial shifter operates at the baud rate. After the RX/TX pin is sampled for the stop bit, the received data in RSR is transferred to the receive data register, if the register is empty. The data which is received on the external RX/TX input pin is sampled three times by a majority detect circuit to determine the logic level that has been placed onto the RX/TX pin. It should be noted that the RSR register, unlike many other registers, is not directly mapped into the Data Memory area and as such is not available to the application program for direct read/write operations.

Receiving Data

When the UART receiver is receiving data, the data is serially shifted in on the external RX/TX input pin, LSB first. In the read mode, the TXR_RXR register forms a buffer between the internal bus and the receiver shift register. The TXR_RXR register is a four byte deep FIFO data buffer, where four bytes can be held in the FIFO while a fifth byte can continue to be received. Note that the application program must ensure that the data is read from TXR_RXR before the fifth byte has been completely shifted in, otherwise this fifth byte will be discarded and an overrun error OERR will be subsequently indicated. For continuous multi-byte data transmission, it is strongly recommended that the receiver uses two stop bits to avoid a receiving error caused by the accumulated error of the receiver baud rate frequency.

The steps to initiate a data transfer can be summarized as follows:

- Make the correct selection of BNO, PRT1~PRT0, PREN and STOPS bits to define the word length and parity type and number of stop bits.
- Setup the BRDH and BRDL registers and the UMOD2~UMOD0 bits to select the desired baud rate.
- Set the RXEN bit to ensure that the RX/TX pin is used as a UART receiver pin.

At this point the receiver will be enabled which will begin to look for a start bit.

When a character is received the following sequence of events will occur:

- The RXIF bit in the USR register will be set when the TXR_RXR register has data available, the number of the available data bytes can be checked by polling the RxCNT register content.
- When the contents of the shift register have been transferred to the TXR_RXR register and Receiver FIFO trigger level is reached if the RIE bit is set, then an interrupt will be generated.
- If during reception, a frame error, noise error, parity error, or an overrun error has been detected, then the error flags can be set.

The RXIF bit can be cleared using the following software sequence:

1. A USR register access
2. A TXR_RXR register read execution

Receiving Break

Any break character received by the UART will be managed as a framing error. The receiver will count and expect a certain number of bit times as specified by the values programmed into the BNO plus one or two stop bits. If the break is much longer than 13 bit times, the reception will be considered as complete after the number of bit times specified by BNO plus one or two stop

bits. The RXIF bit is set, FERR is set, zeros are loaded into the receive data register, interrupts are generated if appropriate and the RIDLE bit is set. A break is regarded as a character that contains only zeros with the FERR flag set. If a long break signal has been detected, the receiver will regard it as a data frame including a start bit, data bits and the invalid stop bit and the FERR flag will be set. The receiver must wait for a valid stop bit before looking for the next start bit. The receiver will not make the assumption that the break condition on the line is the next start bit. The break character will be loaded into the buffer and no further data will be received until one or two stop bits are received. It should be noted that the RIDLE read only flag will go high when the stop bits have not yet been received. The reception of a break character on the UART registers will result in the following:

- The framing error flag, FERR, will be set.
- The receive data register, TXR_RXR, will be cleared.
- The OERR, NF, PERR, RIDLE or RXIF flags will possibly be set.

Idle Status

When the receiver is reading data, which means it will be in between the detection of a start bit and the reading of a stop bit, the receiver status flag in the USR register, otherwise known as the RIDLE flag, will have a zero value. In between the reception of a stop bit and the detection of the next start bit, the RIDLE flag will have a high value, which indicates the receiver is in an idle condition.

Receiver Interrupt

The read only receive interrupt flag RXIF in the USR register is set by an edge generated by the receiver. An interrupt is generated if RIE=1, when a word is transferred from the Receive Shift Register, RSR, to the Receive Data Register, TXR_RXR. An overrun error can also generate an interrupt if RIE=1.

When a subroutine will be called with an execution time longer than the time for UART to receive five data bytes, if the UART received data could not be read in time during the subroutine execution, clear the RXEN bit to zero in advance to suspend data reception. If the UART interrupt could not be served in time to process the overrun error during the subroutine execution, ensure that both EMI and RXEN bits are disabled during this period, and then enable EMI and RXEN again after the subroutine execution has been completed to continue the UART data reception.

Managing Receiver Errors

Several types of reception errors can occur within the UART module, the following section describes the various types and how they are managed by the UART.

Overrun Error – OERR

The TXR_RXR register is composed of a four byte deep FIFO data buffer, where four bytes can be held in the FIFO register, while a fifth byte can continue to be received. Before this fifth byte has been entirely shifted in, the data should be read from the TXR_RXR register. If this is not done, the overrun error flag OERR will be consequently indicated.

In the event of an overrun error occurring, the following will happen:

- The OERR flag in the USR register will be set.
- The TXR_RXR contents will not be lost.
- The shift register will be overwritten.
- An interrupt will be generated if the RIE bit is set.

When the OERR flag is set to "1", it is necessary to read five data bytes from the four-byte deep receiver FIFO and the shift register immediately to avoid unexpected errors, such as the UART is unable to receive data. If such an error occurs, clear the RXEN bit to "0" then set it to "1" again to continue data reception.

The OERR flag can be cleared by an access to the USR register followed by a read to the TXR_RXR register.

Noise Error – NF

Over-sampling is used for data recovery to identify valid incoming data and noise. If noise is detected within a frame the following will occur:

- The read only noise flag, NF, in the USR register will be set on the rising edge of the RXIF bit.
- Data will be transferred from the Shift register to the TXR_RXR register.
- No interrupt will be generated. However this bit rises at the same time as the RXIF bit which itself generates an interrupt.

Note that the NF flag is reset by a USR register read operation followed by a TXR_RXR register read operation.

Framing Error – FERR

The read only framing error flag, FERR, in the USR register, is set if a zero is detected instead of stop bits. If two stop bits are selected, both stop bits must be high; otherwise the FERR flag will be set. The FERR flag and the received data will be recorded in the USR and TXR_RXR registers respectively, and the flag is cleared in any reset.

Parity Error – PERR

The read only parity error flag, PERR, in the USR register, is set if the parity of the received word is incorrect. This error flag is only applicable if the parity is enabled, PREN=1, and if the parity type, odd, even, mark or space, is selected. The read only PERR flag and the received data will be recorded in the USR and TXR_RXR registers respectively. It is cleared on any reset, it should be noted that the flags, FERR and PERR, in the USR register should first be read by the application program before reading the data word.

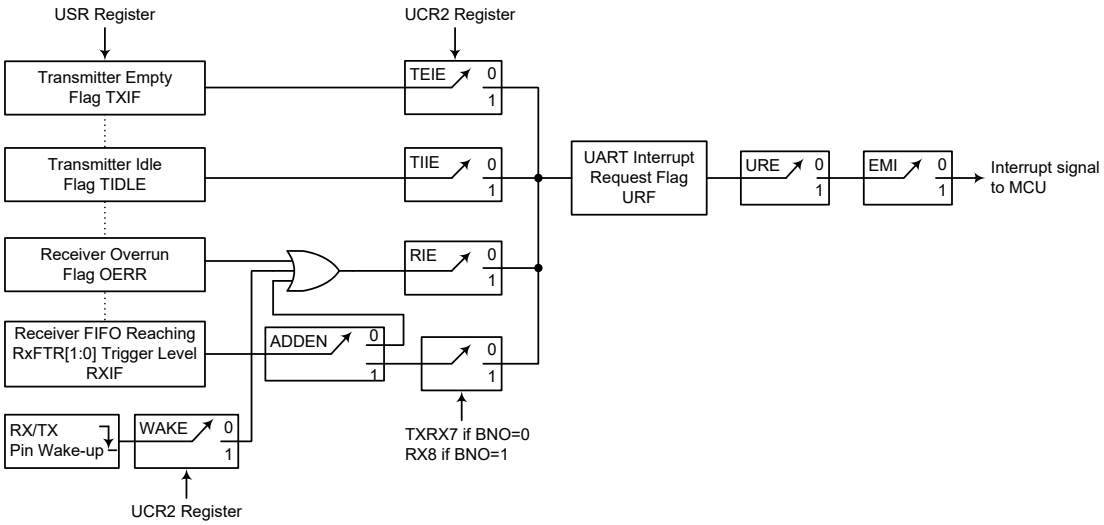
UART Interrupt Structure

Several individual UART conditions can generate a UART interrupt. When these conditions exist, a low pulse will be generated to get the attention of the microcontroller. These conditions are a transmitter data register empty, transmitter idle, receiver reaching FIFO trigger level, receiver overrun, address detect and an RX/TX pin wake-up. When any of these conditions are created, if the global interrupt enable bit and its corresponding interrupt control bit are enabled and the stack is not full, the program will jump to its corresponding interrupt vector where it can be serviced before returning to the main program. Four of these conditions have the corresponding USR register flags which will generate a UART interrupt if its associated interrupt enable control bit in the UCR2 register is set. The two transmitter interrupt conditions have their own corresponding enable control bits, while the two receiver interrupt conditions have a shared enable control bit. These enable bits can be used to mask out individual UART interrupt sources.

The address detect condition, which is also a UART interrupt source, does not have an associated flag, but will generate a UART interrupt when an address detect condition occurs if its function is enabled by setting the ADDEN bit in the UCR2 register. An RX/TX pin wake-up, which is also a UART interrupt source, does not have an associated flag, but will generate a UART interrupt if the

UART clock (f_H) source is switched off and the WAKE and RIE bits in the UCR2 register are set when a falling edge on the RX/TX pin occurs.

Note that the USR register flags are read only and cannot be cleared or set by the application program, neither will they be cleared when the program jumps to the corresponding interrupt servicing routine, as is the case for some of the other interrupts. The flags will be cleared automatically when certain actions are taken by the UART, the details of which are given in the UART register section. The overall UART interrupt can be disabled or enabled by the related interrupt enable control bits in the interrupt control registers of the microcontroller to decide whether the interrupt requested by the UART module is masked out or allowed.



UART Interrupt Structure

Address Detect Mode

Setting the Address Detect Mode bit, ADDEN, in the UCR2 register, enables this special mode. If this bit is enabled then an additional qualifier will be placed on the generation of a Receiver Data Available interrupt, which is requested by the RXIF flag. If the ADDEN bit is enabled, then when data is available, an interrupt will only be generated, if the highest received bit has a high value. Note that the URE and EMI interrupt enable bits must also be enabled for correct interrupt generation. This highest address bit is the 9th bit if BNO=1 or the 8th bit if BNO=0. If this bit is high, then the received word will be defined as an address rather than data. A Data Available interrupt will be generated every time the last bit of the received word is set. If the ADDEN bit is not enabled, then a Receiver Data Available interrupt will be generated each time the RXIF flag is set, irrespective of the data last bit status. The address detect mode and parity enable are mutually exclusive functions. Therefore if the address detect mode is enabled, then to ensure correct operation, the parity function should be disabled by resetting the parity enable bit PREN to zero.

ADDEN	9th Bit if BNO=1 8th Bit if BNO=0	UART Interrupt Generated
0	0	√
	1	√
1	0	×
	1	√

ADDEN Bit Function

UART Power Down and Wake-up

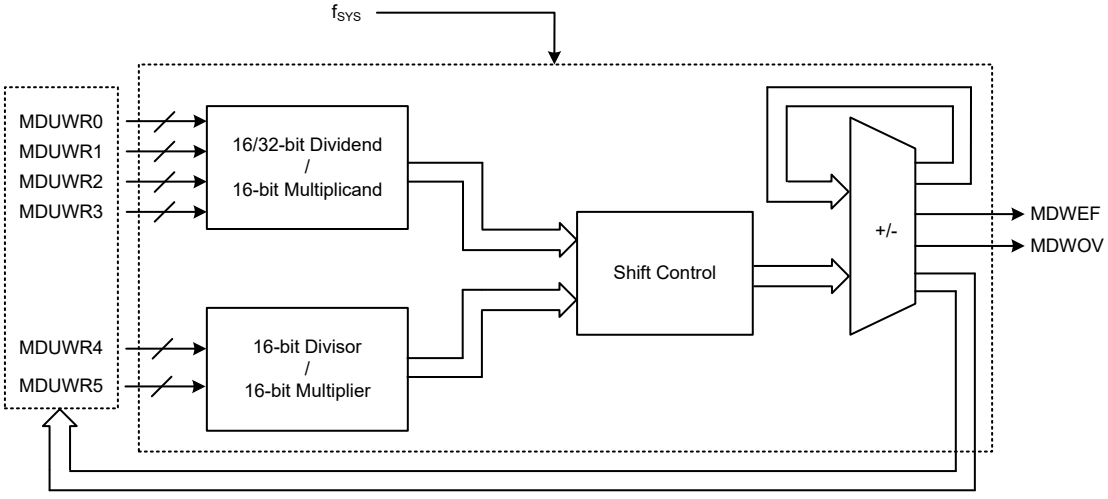
When the UART clock (f_H) is off, the UART will cease to function, all clock sources to the module are shutdown. If the UART clock (f_H) is off while a transmission is still in progress, then the transmission will be paused until the UART clock source derived from the microcontroller is activated. In a similar way, if the MCU enters the IDLE or SLEEP mode while receiving data, then the reception of data will likewise be paused. When the MCU enters the IDLE or SLEEP mode, note that the USR, UCR1, UCR2, UCR3, UFCR, RxCNT, TXR_RXR as well as the BRDH and BRDL registers will not be affected. It is recommended to make sure first that the UART data transmission or reception has been finished before the microcontroller enters the IDLE or SLEEP mode.

The UART function contains a receiver RX/TX pin wake-up function, which is enabled or disabled by the WAKE bit in the UCR2 register. If this bit, along with the UART enable bit, UARTEN, the receiver enable bit, RXEN and the receiver interrupt bit, RIE, are all set when the UART clock (f_H) is off, then a falling edge on the RX/TX pin will trigger an RX/TX pin wake-up UART interrupt. Note that as it takes certain system clock cycles after a wake-up, before normal microcontroller operation resumes, any data received during this time on the RX/TX pin will be ignored.

For a UART wake-up interrupt to occur, in addition to the bits for the wake-up being set, the global interrupt enable bit, EMI, and the UART interrupt enable bit, URE, must be set. If the EMI and URE bits are not set then only a wake up event will occur and no interrupt will be generated. Note also that as it takes certain system clock cycles after a wake-up before normal microcontroller resumes, the UART interrupt will not be generated until after this time has elapsed.

16-bit Multiplication Division Unit – MDU

The device has a 16-bit Multiplication Division Unit, MDU, which integrates a 16-bit unsigned multiplier and a 32-bit/16-bit divider. The MDU, in replacing the software multiplication and division operations, can therefore save large amounts of computing time as well as the Program and Data Memory space. It also reduces the overall microcontroller loading and results in the overall system performance improvements.



16-Bit MDU Block Diagram

MDU Registers

The multiplication and division operations are implemented in a specific way, a specific write access sequence of a series of MDU data registers. The status register, MDUWCTRL, provides the indications for the MDU operation. The data register each is used to store the data regarded as the different operand corresponding to different MDU operations.

Register Name	Bit							
	7	6	5	4	3	2	1	0
MDUWR0	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR1	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR2	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR3	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR4	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR5	D7	D6	D5	D4	D3	D2	D1	D0
MDUWCTRL	MDWEF	MDWOV	—	—	—	—	—	—

MDU Register List

• MDUWRn Register (n=0~5)

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	x	x	x	x	x	x	x	x

"x": unknown

Bit 7~0 **D7~D0:** 16-bit MDU data register n

• MDUWCTRL Register

Bit	7	6	5	4	3	2	1	0
Name	MDWEF	MDWOV	—	—	—	—	—	—
R/W	R	R	—	—	—	—	—	—
POR	0	0	—	—	—	—	—	—

Bit 7 **MDWEF:** 16-bit MDU error flag
0: Normal
1: Abnormal
This bit will be set to 1 if the data register MDUWRn is written or read as the MDU operation is executing. This bit should be cleared to 0 by reading the MDUWCTRL register if it is equal to 1 and the MDU operation is completed.

Bit 6 **MDWOV:** 16-bit MDU overflow flag
0: No overflow occurs
1: Multiplication product > FFFFH or Divisor=0
When an operation is completed, this bit will be updated by hardware to a new value corresponding to the current operation situation.

Bit 5~0 Unimplemented, read as "0"

MDU Operation

For this MDU the multiplication or division operation is carried out in a specific way and is determined by the write access sequence of the six MDU data registers, MDUWR0~MDUWR5. The low byte data, regardless of the dividend, multiplicand, divisor or multiplier, must first be written into the corresponding MDU data register followed by the high byte data. All MDU operations will be executed after the MDUWR5 register is write-accessed together with the correct specific write access sequence of the MDUWRn. Note that it is not necessary to consecutively write data into the MDU data registers but must be in a correct write access sequence. Therefore, a non-write MDUWRn instruction or an interrupt, etc., can be inserted into the correct write access sequence without destroying the write operation. The relationship between the write access sequence and the MDU operation is shown in the following.

- 32-bit/16-bit division operation: Write data sequentially into the six MDU data registers from MDUWR0 to MDUWR5.
- 16-bit/16-bit division operation: Write data sequentially into the specific four MDU data registers in a sequence of MDUWR0, MDUWR1, MDUWR4 and MDUWR5 with no write access to MDUWR2 and MDUWR3.
- 16-bit×16-bit multiplication operation: Write data sequentially into the specific four MDU data register in a sequence of MDUWR0, MDUWR4, MDUWR1 and MDUWR5 with no write access to MDUWR2 and MDUWR3.

After the specific write access sequence is determined, the MDU will start to perform the corresponding operation. The calculation time necessary for these MDU operations are different. During the calculation time any read/write access to the six MDU data registers is forbidden. After the completion of each operation, it is necessary to check the operation status in the MDUWCTRL register to make sure that whether the operation is correct or not. Then the operation result can be read out from the corresponding MDU data registers in a specific read access sequence if the operation is correctly finished. The necessary calculation time for different MDU operations is listed in the following.

- 32-bit/16-bit division operation: $17\times t_{SYS}$.
- 16-bit/16-bit division operation: $9\times t_{SYS}$.
- 16-bit×16-bit multiplication operation: $11\times t_{SYS}$.

The operation results will be stored in the corresponding MDU data registers and should be read out from the MDU data registers in a specific read access sequence after the operation is completed. Note that it is not necessary to consecutively read data out from the MDU data registers but must be in a correct read access sequence. Therefore, a non-read MDUWRn instruction or an interrupt, etc., can be inserted into the correct read access sequence without destroying the read operation. The relationship between the operation result read access sequence and the MDU operation is shown in the following.

- 32-bit/16-bit division operation: Read the quotient from MDUWR0 to MDUWR3 and remainder from MDUWR4 and MDUWR5 sequentially.
- 16-bit/16-bit division operation: Read the quotient from MDUWR0 and MDUWR1 and remainder from MDUWR4 and MDUWR5 sequentially.
- 16-bit×16-bit multiplication operation: Read the product sequentially from MDUWR0 to MDUWR3.

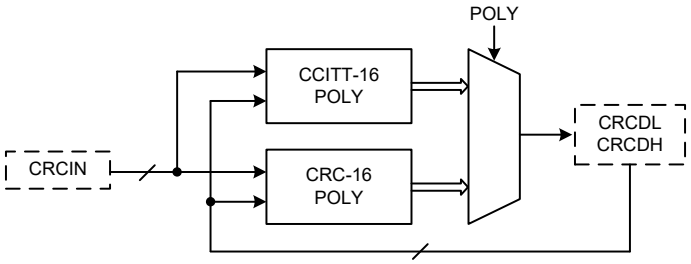
The overall important points for the MDU read/write access sequence and calculation time are summarized in the following table. Note that the device should not enter the IDLE or SLEEP mode until the MDU operation is totally completed, otherwise the MDU operation will fail.

Items \ Operations	32-bit / 16-bit Division	16-bit / 16-bit Division	16-bit × 16-bit Multiplication
Write Sequence First write ↓ ↓ ↓ ↓ Last write	Dividend Byte 0 written to MDUWR0 Dividend Byte 1 written to MDUWR1 Dividend Byte 2 written to MDUWR2 Dividend Byte 3 written to MDUWR3 Divisor Byte 0 written to MDUWR4 Divisor Byte 1 written to MDUWR5	Dividend Byte 0 written to MDUWR0 Dividend Byte 1 written to MDUWR1 Divisor Byte 0 written to MDUWR4 Divisor Byte 1 written to MDUWR5	Multiplicand Byte 0 written to MDUWR0 Multiplier Byte 0 written to MDUWR4 Multiplicand Byte 1 written to MDUWR1 Multiplier Byte 1 written to MDUWR5
Calculation Time	$17\times t_{SYS}$	$9\times t_{SYS}$	$11\times t_{SYS}$
Read Sequence First read ↓ ↓ ↓ ↓ Last read	Quotient Byte 0 read from MDUWR0 Quotient Byte 1 read from MDUWR1 Quotient Byte 2 read from MDUWR2 Quotient Byte 3 read from MDUWR3 Remainder Byte 0 read from MDUWR4 Remainder Byte 1 read from MDUWR5	Quotient Byte 0 read from MDUWR0 Quotient Byte 1 read from MDUWR1 Remainder Byte 0 read from MDUWR4 Remainder Byte 1 read from MDUWR5	Product Byte 0 read from MDUWR0 Product Byte 1 read from MDUWR1 Product Byte 2 read from MDUWR2 Product Byte 3 read from MDUWR3

MDU Operations Summary

Cyclic Redundancy Check – CRC

The Cyclic Redundancy Check, CRC, calculation unit is an error detection technique test algorithm and uses to verify data transmission or storage data correctness. A CRC calculation takes a data stream or a block of data as input and generates a 16-bit output remainder. Ordinarily, a data stream is suffixed by a CRC code and used as a checksum when being sent or stored. Therefore, the received or restored data stream is calculated by the same generator polynomial as described in the following section.



CRC Block Diagram

CRC Registers

The CRC generator contains an 8-bit CRC data input register, CRCIN, and a CRC checksum register pair, CRCDH and CRCDL. The CRCIN register is used to input new data and the CRCDH and CRCDL registers are used to hold the previous CRC calculation result. A CRC control register, CRCCR, is used to select which CRC generating polynomial is used.

Register Name	Bit							
	7	6	5	4	3	2	1	0
CRCCR	—	—	—	—	—	—	—	POLY
CRCIN	D7	D6	D5	D4	D3	D2	D1	D0
CRCDL	D7	D6	D5	D4	D3	D2	D1	D0
CRCDH	D7	D6	D5	D4	D3	D2	D1	D0

CRC Register List

• **CRCCR Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	—	POLY
R/W	—	—	—	—	—	—	—	R/W
POR	—	—	—	—	—	—	—	0

Bit 7~1 Unimplemented, read as "0"

Bit 0 **POLY:** 16-bit CRC generating polynomial selection
0: CRC-CCITT: $X^{16}+X^{12}+X^5+1$
1: CRC-16: $X^{16}+X^{15}+X^2+1$

• **CRCIN Register**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0:** CRC input data register

• **CRCDL Register**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0:** 16-bit CRC checksum low byte data register

• **CRCDH Register**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0:** 16-bit CRC checksum high byte data register

CRC Operation

The CRC generator provides the 16-bit CRC result calculation based on the CRC16 and CCITT CRC16 polynomials. In this CRC generator, there are only these two polynomials available for the numeric values calculation. It cannot support the 16-bit CRC calculations based on any other polynomials.

The following two expressions can be used for the CRC generating polynomial which is determined using the POLY bit in the CRC control register, CRCCR. The CRC calculation result is called as the CRC checksum, CRCSUM, and stored in the CRC checksum register pair, CRCDH and CRCDL.

- CRC-CCITT: $X^{16} + X^{12} + X^5 + 1$.
- CRC-16: $X^{16} + X^{15} + X^2 + 1$.

CRC Computation

Each write operation to the CRCIN register creates a combination of the previous CRC value stored in the CRCDH and CRCDL registers and the new data input. The CRC unit calculates the CRC data register value is based on byte by byte. It will take one MCU instruction cycle to calculate the CRC checksum.

CRC Calculation Procedures

1. Clear the checksum register pair, CRCDH and CRCDL.
2. Execute an "Exclusive OR" operation with the 8-bit input data byte and the 16-bit CRCSUM high byte. The result is called the temporary CRCSUM.
3. Shift the temporary CRCSUM value left by one bit and move a "0" into the LSB.
4. Check the shifted temporary CRCSUM value after procedure 3.

   If the MSB is 0, then this shifted temporary CRCSUM will be considered as a new temporary CRCSUM.

   Otherwise, execute an "Exclusive OR" operation with the shifted temporary CRCSUM in procedure 3 and a data "8005H". Then the operation result will be regarded as the new temporary CRCSUM.

   Note that the data to be perform an "Exclusive OR" operation is "8005H" for the CRC-16 polynomial while for the CRC-CCITT polynomial the data is "1021H".
5. Repeat the procedure 3 ~ procedure 4 until all bits of the input data byte are completely calculated.
6. Repeat the procedure 2 ~ procedure 5 until all of the input data bytes are completely calculated. Then, the latest calculated result is the final CRC checksum, CRCSUM.

CRC Calculation Examples:

- Write 1 byte input data into the CRCIN register and the corresponding CRC checksum are individually calculated as the following table shown.

CRC Data Input / CRC Polynomial	00H	01H	02H	03H	04H	05H	06H	07H
CRC-CCITT ($X^{16}+X^{12}+X^5+1$)	0000H	1021H	2042H	3063H	4084H	50A5H	60C6H	70E7H
CRC-16 ($X^{16}+X^{15}+X^2+1$)	0000H	8005H	800FH	000AH	801BH	001EH	0014H	8011H

Note: The initial value of the CRC checksum register pair, CRCDH and CRCDL, is zero before each CRC input data is written into the CRCIN register.

- Write 4 bytes input data into the CRCIN register sequentially and the CRC checksum are sequentially listed in the following table.

CRC Data Input / CRC Polynomial	CRCIN=78H→56H→34H→12H
CRC-CCITT ($X^{16}+X^{12}+X^5+1$)	(CRCDH, CRCDL)=FF9FH→BBC3H→A367H→D0FAH
CRC-16 ($X^{16}+X^{15}+X^2+1$)	(CRCDH, CRCDL)=0110h→91F1h→F2DEh→5C43h

Note: The initial value of the CRC checksum register pair, CRCDH and CRCDL, is zero before the sequential CRC data input operation.

Program Memory CRC Checksum Calculation Example

1. Clear the checksum register pair, CRCDH and CRCDL.
2. Select the CRC-CCITT or CRC-16 polynomial as the generating polynomial using the POLY bit in the CRCCR register.
3. Execute the table read instruction to read the program memory data value.
4. Write the table data low byte into the CRCIN register and execute the CRC calculation with the current CRCSUM value. Then a new CRCSUM result will be obtained and stored in the CRC checksum register pair, CRCDH and CRCDL.
5. Write the table data high byte into the CRCIN register and execute the CRC calculation with the current CRCSUM value. Then a new CRCSUM result will be obtained and stored in the CRC checksum register pair, CRCDH and CRCDL.

6. Repeat the procedure 3 ~ procedure 5 to read the next program memory data value and execute the CRC calculation until all program memory data are read followed by the sequential CRC calculation. Then the value in the CRC checksum register pair is the final CRC calculation result.

Low Voltage Detector – LVD

The device has a Low Voltage Detector function, also known as LVD. This enables the device to monitor the power supply voltage, V_{DD}, and provide a warning signal should it fall below a certain level. This function may be especially useful in battery applications where the supply voltage will gradually reduce as the battery ages, as it allows an early warning battery low signal to be generated. The Low Voltage Detector also has the capability of generating an interrupt signal.

LVD Register

The Low Voltage Detector function is controlled using a single register with the name LVDC. Three bits in this register, VLVD2~VLVD0, are used to select one of eight fixed voltages below which a low voltage condition will be determined. A low voltage condition is indicated when the LVDO bit is set. If the LVDO bit is low, this indicates that the V_{DD} voltage is above the preset low voltage value. The LVDEN bit is used to control the overall on/off function of the low voltage detector. Setting the bit high will enable the low voltage detector. Clearing the bit to zero will switch off the internal low voltage detector circuits. As the low voltage detector will consume a certain amount of power, it may be desirable to switch off the circuit when not in use, an important consideration in power sensitive battery powered applications.

- **LVDC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	LVDO	LVDEN	—	VLVD2	VLVD1	VLVD0
R/W	—	—	R	R/W	—	R/W	R/W	R/W
POR	—	—	0	0	—	0	0	0

Bit 7~6 Unimplemented, read as “0”

Bit 5 **LVDO:** LVD output flag
0: No Low Voltage Detected
1: Low Voltage Detected

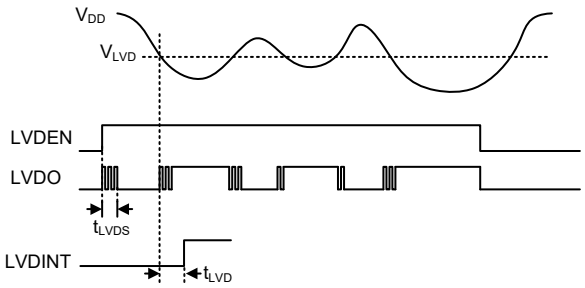
Bit 4 **LVDEN:** Low voltage detector enable control
0: Disable
1: Enable

Bit 3 Unimplemented, read as “0”

Bit 2~0 **VLVD2~VLVD0:** LVD voltage selection
000: 2.0V
001: 2.2V
010: 2.4V
011: 2.7V
100: 3.0V
101: 3.3V
110: 3.6V
111: 4.0V

LVD Operation

The Low Voltage Detector function operates by comparing the power supply voltage, V_{DD}, with a pre-specified voltage level stored in the LVDC register. This has a range of between 2.0V and 4.0V. When the power supply voltage, V_{DD}, falls below this pre-determined value, the LVDO bit will be set high indicating a low power supply voltage condition. The Low Voltage Detector function is supplied by a reference voltage which will be automatically enabled. When the device enters the SLEEP Mode, the low voltage detector will automatically be disabled even if the LVDEN bit is high. After enabling the Low Voltage Detector, a time delay t_{LVDS} should be allowed for the circuitry to stabilise before reading the LVDO bit. Note also that as the V_{DD} voltage may rise and fall rather slowly, at the voltage nears that of V_{LVD}, there may be multiple bit LVDO transitions.



LVD Operation

The Low Voltage Detector also has its own interrupt, providing an alternative means of low voltage detection, in addition to polling the LVDO bit. The interrupt will only be generated after a delay of t_{LVD} after the LVDO bit has been set high by a low voltage condition. In this case, the LVF interrupt request flag will be set, causing an interrupt to be generated if V_{DD} falls below the preset LVD voltage. This will cause the device to wake up from the IDLE Mode. However, if the Low Voltage Detector wake up function is not required, then the LVF flag should be first set high before the device enters the IDLE Mode.

PD PHY

Power Delivery, known as PD, is a USB charging standard and technology released by the USB-IF. It uses Type-C interface to implement PD fast charging. With a power output of up to 240W, it can charge mobile phones, tablet and notebook computers. The high wattage output can even provide power for monitors, televisions, etc., and the Type-C interface can support two-way charging, making it more flexible in use.

The device contains a transceiver control circuit compliant with PD standard. With integrated USB BMC encoding/decoding circuit, it can support Dual Role Port operation, PD 3.1 for allowing up to 240W power supply and Programmable Power Supply, PPS. The CC1 and CC2 pins is used for Type-C identification and PD communication transmission, providing VCONN power for E-Mark cable. These two pins contain an over voltage protection circuit, which will disconnect the channel to protect the internal circuit when the voltage is too high. There are two HVO output pins used for the external MOS component on/off control. The PD transceiver control circuit contains an independent control register space. The device writes or reads PD register data using the I^2C interface (master mode) to implement USB Type-C PD communication transmission.

The device has six PD modes. Users can select a PD mode according to their requirements and turn on Type-C detection to start PD communication. When an external PD device is connected, the Type-C identification will automatically be executed. After the identification is completed, an interrupt signal will be generated to inform the device. The device will read data from PD registers

using the I²C interface and then determine to carry out PD charging or discharging transmission according to the identification result. The PD charge/discharge communication is transmitted and received by MCU writing or reading PD register TX/RX buffer data via the I²C interface. The PD protocol communication process is implemented in accordance with PD specification content stipulated by the USB-IF.

PD PHY Registers Type

Bit Type	Definition
RO	Read only
WC	Write with 1 which in automatically cleared by hardware
RW	Readable and writeable
RWH	Readable and writable which can be overwritten by hardware
RC	Readable interrupt bits, cleared on read

PD PHY Registers

Overall operation of the PD PHY is controlled using a series of registers.

Register Name	Bit							
	7	6	5	4	3	2	1	0
VERSION	INT_RES	—	—	—	—	—	—	—
USB_C_CTL1	DRP_TOGGL7	DRP_TOGGL6	DRP_TOGGL5	CURR_SRC4	CURR_SRC3	MODES2	MODES1	MODES0
USB_C_CTL2	UNSUP_ACC	TRY_SRC	ATT_SRC	ATT_SNK	ERR_REC	DIS_ST	UNATT_SRC	UNATT_SNK
USB_C_CTL3	—	—	VBUS_IGNORE	—	—	RESETPHY	—	DET_DIS
CC1_CONTROL	VBUSOK	—	—	—	PDWN1	TXE1	VCONN1	PU1
CC2_CONTROL	VBUSOK	—	—	—	PDWN2	TXE2	VCONN2	PU2
CC_SEL	—	—	—	—	VCONN_SWAP_OFF	VCONN_SWAP_ON	CC_SEL1	CC_SEL0
USB_C_STATUS1	TYPE_C_DET7	TYPE_C_DET6	CC_ORIENT5	CC_ORIENT4	TYPE_C_RSLT3	TYPE_C_RSLT2	TYPE_C_RSLT1	TYPE_C_RSLT0
USB_C_STATUS2	—	—	VBUS_REQ	PD_NOT_ALLOWED	—	—	OVRTEMP	SHORT
USB_C_STATUS3	TYPEC_ACTIVE	—	—	—	—	—	—	—
CC1_CMP	—	—	—	DET_3A	DET_1P5A	DET_DEF	DET_RD	DET_RA
CC2_CMP	—	—	—	DET_3A	DET_1P5A	DET_DEF	DET_RD	DET_RA
CC1_STATUS	—	SRC_RX16	SRC_RX15	SRC_RP1	PWR3A_SNK1	PWR1P5A_SNK1	PWRDEF_SNK1	SNK_RP1
CC2_STATUS	—	SRC_RX26	SRC_RX25	SRC_RP2	PWR3A_SNK2	PWR1P5A_SNK2	PWRDEF_SNK2	SNK_RP2
VBUS_MON	VBUS_MON_EN	COMP	DAC5	DAC4	DAC3	DAC2	DAC1	DAC0
AFE_TRIM2	—	—	—	—	—	—	TRIM_CCDRV1	TRIM_CCDRV0
AFE_TRIM3	—	TRIM_SLICE2	TRIM_SLICE1	TRIM_SLICE0	—	—	—	—
POWER	VCONN_CTRL_EXT	—	—	—	PWR3	PWR2	PWR1	PWR0
IRQ1	—	—	—	—	I_VBUS_DROP	—	I_OVRTEMP	I_SHORT
IRQ2	I_CC_CHANGE	I_PD_RX	I_PD_HR_RX	I_PD_CR_RX	I_PD_TX_OK	I_PD_TX_FAIL	I_FAST_SWAP	I_TX_DISCARD

Register Name	Bit							
	7	6	5	4	3	2	1	0
IRQ_MSK1	—	—	—	—	M_VBUS_DROP	—	M_OVR-TEMP	M_SHORT
IRQ_MSK2	M_CC_CHANGE	M_PD_RX	M_PD_HR_RX	M_PD_CR_RX	M_PD_TX_OK	M_PD_TX_FAIL	M_FAST_SWAP	—
PD_CFG1	ID_INSERT	VBUS_HIGH_VOLT	GUIDE_TRY_SNK	—	RESET_MSG_ID	SOP_TO_RESET2	SOP_TO_RESET1	SOP_TO_RESET0
PD_CFG2	FAST_SWAP_SNK	FAST_SWAP_SRC	CDR_SE-LECT	SOP_RCV4	SOP_RCV3	SOP_RCV2	SOP_RCV1	SOP_RCV0
PD_CFG3	—	—	P_DATA_ROLE_DP	P_PWR_ROLE_DP	P_DATA_ROLE_PR	P_PWR_ROLE_PR	P_DATA_ROLE_SOP	P_PWR_ROLE_SOP
SHORT_PRO-TECT	SHORT_RESET	SHORT_TIME6	SHORT_TIME5	SHORT_TIME4	SHORT_TIME3	SHORT_TIME2	SHORT_TIME1	SHORT_TIME0
PD_STATUS	FAST_SWAP	—	RX_RE-SULT5	RX_RE-SULT4	RX_RE-SULT3	TX_RE-SULT2	TX_RE-SULT1	TX_RE-SULT0
RX_STATUS	RX_DATA	RX_OVER-RUN	—	—	—	—	—	RX_CLEAR
RX_INFO	RX_BYTES7	RX_BYTES6	RX_BYTES5	RX_BYTES4	RX_BYTES3	RX_SOP2	RX_SOP1	RX_SOP0
TX_COM-MAND	TX_CMD7	TX_CMD6	TX_CMD5	—	—	TX_WAIT_RP	TX_START	TXBUF_READY
TX_INFO	—	—	TX_RE-TRIES5	TX_RE-TRIES4	TX_RE-TRIES3	TX_SOP2	TX_SOP1	TX_SOP0
RX_PACKET_DATA61	D7	D6	D5	D4	D3	D2	D1	D0
RX_PACKET_DATA62	D7	D6	D5	D4	D3	D2	D1	D0
RX_PACKET_DATA63	D7	D6	D5	D4	D3	D2	D1	D0
RX_PACKET_DATA64	D7	D6	D5	D4	D3	D2	D1	D0
RX_PACKET_DATA65	D7	D6	D5	D4	D3	D2	D1	D0
RX_PACKET_DATA66	D7	D6	D5	D4	D3	D2	D1	D0
RX_PACKET_DATA67	D7	D6	D5	D4	D3	D2	D1	D0
RX_PACKET_DATA68	D7	D6	D5	D4	D3	D2	D1	D0
RX_PACKET_DATA69	D7	D6	D5	D4	D3	D2	D1	D0
RX_PACKET_DATA70	D7	D6	D5	D4	D3	D2	D1	D0
RX_PACKET_DATA71	D7	D6	D5	D4	D3	D2	D1	D0
RX_PACKET_DATA72	D7	D6	D5	D4	D3	D2	D1	D0
RX_PACKET_DATA73	D7	D6	D5	D4	D3	D2	D1	D0
RX_PACKET_DATA74	D7	D6	D5	D4	D3	D2	D1	D0
RX_PACKET_DATA75	D7	D6	D5	D4	D3	D2	D1	D0
RX_PACKET_DATA76	D7	D6	D5	D4	D3	D2	D1	D0
RX_PACKET_DATA77	D7	D6	D5	D4	D3	D2	D1	D0
RX_PACKET_DATA78	D7	D6	D5	D4	D3	D2	D1	D0
RX_PACKET_DATA79	D7	D6	D5	D4	D3	D2	D1	D0

Register Name	Bit							
	7	6	5	4	3	2	1	0
RX_PACKET_DATA80	D7	D6	D5	D4	D3	D2	D1	D0
RX_PACKET_DATA81	D7	D6	D5	D4	D3	D2	D1	D0
RX_PACKET_DATA82	D7	D6	D5	D4	D3	D2	D1	D0
RX_PACKET_DATA83	D7	D6	D5	D4	D3	D2	D1	D0
RX_PACKET_DATA84	D7	D6	D5	D4	D3	D2	D1	D0
RX_PACKET_DATA85	D7	D6	D5	D4	D3	D2	D1	D0
RX_PACKET_DATA86	D7	D6	D5	D4	D3	D2	D1	D0
RX_PACKET_DATA87	D7	D6	D5	D4	D3	D2	D1	D0
RX_PACKET_DATA88	D7	D6	D5	D4	D3	D2	D1	D0
RX_PACKET_DATA89	D7	D6	D5	D4	D3	D2	D1	D0
RX_PACKET_DATA90	D7	D6	D5	D4	D3	D2	D1	D0
TX_PACKET_DATA91	D7	D6	D5	D4	D3	D2	D1	D0
TX_PACKET_DATA92	D7	D6	D5	D4	D3	D2	D1	D0
TX_PACKET_DATA93	D7	D6	D5	D4	D3	D2	D1	D0
TX_PACKET_DATA94	D7	D6	D5	D4	D3	D2	D1	D0
TX_PACKET_DATA95	D7	D6	D5	D4	D3	D2	D1	D0
TX_PACKET_DATA96	D7	D6	D5	D4	D3	D2	D1	D0
TX_PACKET_DATA97	D7	D6	D5	D4	D3	D2	D1	D0
TX_PACKET_DATA98	D7	D6	D5	D4	D3	D2	D1	D0
TX_PACKET_DATA99	D7	D6	D5	D4	D3	D2	D1	D0
TX_PACKET_DATA100	D7	D6	D5	D4	D3	D2	D1	D0
TX_PACKET_DATA101	D7	D6	D5	D4	D3	D2	D1	D0
TX_PACKET_DATA102	D7	D6	D5	D4	D3	D2	D1	D0
TX_PACKET_DATA103	D7	D6	D5	D4	D3	D2	D1	D0
TX_PACKET_DATA104	D7	D6	D5	D4	D3	D2	D1	D0
TX_PACKET_DATA105	D7	D6	D5	D4	D3	D2	D1	D0
TX_PACKET_DATA106	D7	D6	D5	D4	D3	D2	D1	D0
TX_PACKET_DATA107	D7	D6	D5	D4	D3	D2	D1	D0
TX_PACKET_DATA108	D7	D6	D5	D4	D3	D2	D1	D0

Register Name	Bit							
	7	6	5	4	3	2	1	0
TX_PACKET_DATA109	D7	D6	D5	D4	D3	D2	D1	D0
TX_PACKET_DATA110	D7	D6	D5	D4	D3	D2	D1	D0
TX_PACKET_DATA111	D7	D6	D5	D4	D3	D2	D1	D0
TX_PACKET_DATA112	D7	D6	D5	D4	D3	D2	D1	D0
TX_PACKET_DATA113	D7	D6	D5	D4	D3	D2	D1	D0
TX_PACKET_DATA114	D7	D6	D5	D4	D3	D2	D1	D0
TX_PACKET_DATA115	D7	D6	D5	D4	D3	D2	D1	D0
TX_PACKET_DATA116	D7	D6	D5	D4	D3	D2	D1	D0
TX_PACKET_DATA117	D7	D6	D5	D4	D3	D2	D1	D0
TX_PACKET_DATA118	D7	D6	D5	D4	D3	D2	D1	D0
TX_PACKET_DATA119	D7	D6	D5	D4	D3	D2	D1	D0
TX_PACKET_DATA120	D7	D6	D5	D4	D3	D2	D1	D0
C_OVP	—	—	—	—	ENP2	ENP1	STP2	STP1

PD PHY Register List

The following table indicates the way in which the PD PHY internal registers are affected after a power-on reset occurs.

Address	Register	Power On Reset
00H	VERSION	0 - - - - - - - -
01H	USB_C_CTL1	0 0 0 0 1 0 0 0
02H	USB_C_CTL2	0 0 0 0 0 0 0 0
03H	USB_C_CTL3	- - 0 - - 0 - 0
04H	CC1_CONTROL	1 - - - 1 0 0 0
05H	CC2_CONTROL	1 - - - 1 0 0 0
06H	CC_SEL	- - - - 0 0 0 0
07H	USB_C_STATUS1	0 0 0 0 0 0 0 0
08H	USB_C_STATUS2	- - 0 0 - - 0 0
09H	USB_C_STATUS3	1 - - - - - - -
0AH	CC1_CMP	- - - 0 0 0 0 0
0BH	CC2_CMP	- - - 0 0 0 0 0
0CH	CC1_STATUS	- 0 0 0 0 0 0 0
0DH	CC2_STATUS	- 0 0 0 0 0 0 0
0FH	VBUS_MON	0 0 0 0 0 0 0 0
11H	AFE_TRIM2	- - - - - - 1 0
12H	AFE_TRIM3	- 0 0 0 - - - -
14H	POWER	0 - - - 1 1 1 1
16H	IRQ1	- - - - 0 - 0 0
17H	IRQ2	1 0 0 0 0 0 0 0
18H	IRQ_MSK1	- - - - 0 - 0 0
19H	IRQ_MSK2	0 0 0 0 0 0 0 -

Address	Register	Power On Reset
1AH	PD_CFG1	000- 0000
1BH	PD_CFG2	0000 0000
1CH	PD_CFG3	--00 0000
1DH	SHORT_PROTECT	0010 1001
1EH	PD_STATUS	0-00 0000
1FH	RX_STATUS	00-- ---0
20H	RX_INFO	0000 0000
21H	TX_COMMAND	000- -001
22H	TX_INFO	--01 1000
3DH	RX_PACKET_DATA61	0000 0000
3EH	RX_PACKET_DATA62	0000 0000
3FH	RX_PACKET_DATA63	0000 0000
40H	RX_PACKET_DATA64	0000 0000
41H	RX_PACKET_DATA65	0000 0000
42H	RX_PACKET_DATA66	0000 0000
43H	RX_PACKET_DATA67	0000 0000
44H	RX_PACKET_DATA68	0000 0000
45H	RX_PACKET_DATA69	0000 0000
46H	RX_PACKET_DATA70	0000 0000
47H	RX_PACKET_DATA71	0000 0000
48H	RX_PACKET_DATA72	0000 0000
49H	RX_PACKET_DATA73	0000 0000
4AH	RX_PACKET_DATA74	0000 0000
4BH	RX_PACKET_DATA75	0000 0000
4CH	RX_PACKET_DATA76	0000 0000
4DH	RX_PACKET_DATA77	0000 0000
4EH	RX_PACKET_DATA78	0000 0000
4FH	RX_PACKET_DATA79	0000 0000
50H	RX_PACKET_DATA80	0000 0000
51H	RX_PACKET_DATA81	0000 0000
52H	RX_PACKET_DATA82	0000 0000
53H	RX_PACKET_DATA83	0000 0000
54H	RX_PACKET_DATA84	0000 0000
55H	RX_PACKET_DATA85	0000 0000
56H	RX_PACKET_DATA86	0000 0000
57H	RX_PACKET_DATA87	0000 0000
58H	RX_PACKET_DATA88	0000 0000
59H	RX_PACKET_DATA89	0000 0000
5AH	RX_PACKET_DATA90	0000 0000
5BH	TX_PACKET_DATA91	0000 0000
5CH	TX_PACKET_DATA92	0000 0000
5DH	TX_PACKET_DATA93	0000 0000
5EH	TX_PACKET_DATA94	0000 0000
5FH	TX_PACKET_DATA95	0000 0000
60H	TX_PACKET_DATA96	0000 0000
61H	TX_PACKET_DATA97	0000 0000
62H	TX_PACKET_DATA98	0000 0000
63H	TX_PACKET_DATA99	0000 0000
64H	TX_PACKET_DATA100	0000 0000

Address	Register	Power On Reset
65H	TX_PACKET_DATA101	0000 0000
66H	TX_PACKET_DATA102	0000 0000
67H	TX_PACKET_DATA103	0000 0000
68H	TX_PACKET_DATA104	0000 0000
69H	TX_PACKET_DATA105	0000 0000
6AH	TX_PACKET_DATA106	0000 0000
6BH	TX_PACKET_DATA107	0000 0000
6CH	TX_PACKET_DATA108	0000 0000
6DH	TX_PACKET_DATA109	0000 0000
6EH	TX_PACKET_DATA110	0000 0000
6FH	TX_PACKET_DATA111	0000 0000
70H	TX_PACKET_DATA112	0000 0000
71H	TX_PACKET_DATA113	0000 0000
72H	TX_PACKET_DATA114	0000 0000
73H	TX_PACKET_DATA115	0000 0000
74H	TX_PACKET_DATA116	0000 0000
75H	TX_PACKET_DATA117	0000 0000
76H	TX_PACKET_DATA118	0000 0000
77H	TX_PACKET_DATA119	0000 0000
78H	TX_PACKET_DATA120	0000 0000
7DH	C_OVP	---- 0000

• VERSION Register

Bit	7	6	5	4	3	2	1	0
Name	INT_RES	—	—	—	—	—	—	—
R/W	R/W	—	—	—	—	—	—	—
POR	0	—	—	—	—	—	—	—

Bit 7 **INT_RES:** Interrupts Reset control
0: Reset interrupts by reading
1: Reset interrupts by writing "1" to the bits to be cleared

Bit 6~0 Unimplemented, read as "0"

• USB_C_CTL1 Register

Bit	7	6	5	4	3	2	1	0
Name	DRP_ TOGGL7	DRP_ TOGGL6	DRP_ TOGGL5	CURR_ SRC4	CURR_ SRC3	MODES2	MODES1	MODES0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	1	0	0	0

Bit 7~5 **DRP_TOGGL7~DRP_TOGGL5:** Type-C State Machine configuration
000: 50% in Unattached.SNK State and 50% in Unattached.SRC
001: 40% in Unattached.SNK State and 60% in Unattached.SRC
010: 30% in Unattached.SNK State and 70% in Unattached.SRC
011: Reserved
100: Reserved
101: 60% in Unattached.SNK State and 40% in Unattached.SRC
110: 70% in Unattached.SNK State and 30% in Unattached.SRC
111: Pseudo randomly change the ratio between Unattached.SNK and Unattached.SRC between 34% and 66%

Bit 4~3 **CURR_SRC4~CURR_SRC3:** R_P pull-up current selection
00: No current
01: 80μA – Default current capability
10: 180μA – 1.5A current capability
11: 330μA – 3A current capability

Bit 2~0 **MODES2~MODE0:** PD mode selection
000: SNK (start in sink mode. No Accessory support.)
001: SNK + Accessory Support
010: SRC (start in source mode. No Accessory support.)
011: SRC + Accessory Support
100: DRP (dual role port – will toggle (refer to the DRP_TOGGL[7:5] bits)
101: DRP + Accessory Support
110: Reserved
111: Reserved

Note: This bit field selects which State Machine is used. The ATT_SRC and ATT_SNK bits can be used to switch state machines between source and sink during PR-swap (refer to the USB_C_CTL2 register).

• USB_C_CTL2 Register

Bit	7	6	5	4	3	2	1	0
Name	UNSUP_ACC	TRY_SRC	ATT_SRC	ATT_SNK	ERR_REC	DIS_ST	UNATT_SRC	UNATT_SNK
R/W	WC	R/W	WC	WC	WC	WC	WC	WC
POR	0	0	0	0	0	0	0	0

Bit 7 **UNSUP_ACC:** Unsupported Accessory control
0: No effect
1: Unsupported Accessory – use if software does not know how to support this accessory

When this bit is set high, go to the Unsupported Accessory state if the software does not support this powered accessory.

Bit 6 **TRY_SRC:** TRY.SRC mode control
0: No effect
1: TRY.SRC mode enabled

Bit 5 **ATT_SRC:** Attached.SRC control
0: No effect
1: Set Type-C circuit in Attached.SRC State

When this bit is set high, go to Attached.SRC state, done with a Power Role Swap and valid only from Attached.SNK state.

Bit 4 **ATT_SNK:** Attached.SNK control
0: No effect
1: Set Type-C circuit in Attached.SNK State

When this bit is set high, go to Attached.SNK state, done with a Power Role Swap and valid only from Attached.SRC state.

Bit 3 **ERR_REC:** Type-C circuit ErrorRecovery State control
0: No effect
1: Set Type-C circuit in ErrorRecovery State

When this bit is set high, go to ErrorRecovery state, valid from any state.

Bit 2 **DIS_ST:** Type-C circuit Disabled State control
0: No effect
1: Set Type-C circuit in Disabled State

When this bit is set high, go to Disabled state, valid from any state.

Bit 1 **UNATT_SRC:** Type-C circuit Unattached.SRC State control
0: No effect
1: Set Type-C circuit in Unattached.SRC State – provided the FSM is in an appropriate state

Bit 0 **UNATT_SNK:** Type-C circuit Unattached.SNK State control
0: No effect
1: Set Type-C circuit in Unattached.SNK State – provided the FSM is in an appropriate state

• **USB_C_CTL3 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	VBUS_IGNORE	—	—	RESETPHY	—	DET_DIS
R/W	—	—	R/W	—	—	WC	—	R/W
POR	—	—	0	—	—	0	—	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **VBUS_IGNORE:** VBUS ignore selection
0: Use VBUS for state change
1: Ignore VBUS during PR-swap

Bit 4~3 Unimplemented, read as "0"

Bit 2 **RESETPHY:** PD PHY logic reset control
0: Normal operation
1: Force reset of PD logic - Does not reset registers

Bit 1 Unimplemented, read as "0"

Bit 0 **DET_DIS:** Type-C detection control
0: Type-C detection enabled
1: Type-C detection disabled - Control only by software, ignore FSM

• **CC1_CONTROL Registe**

Bit	7	6	5	4	3	2	1	0
Name	VBUSOK	—	—	—	PDWN1	TXE1	VCONN1	PU1
R/W	RO	—	—	—	RWH	RWH	RWH	RWH
POR	1	—	—	—	1	0	0	0

Bit 7 **VBUSOK:** VBUS active status
0: VBUS is not active
1: VBUS is active

Bit 6~4 Unimplemented, read as "0"

Bit 3 **PDWN1:** 5.1kΩ pull down resistor to CC1 control
0: Disable
1: Enable

Bit 2 **TXE1:** CC1 PD driver control
0: Disable
1: Enable

Bit 1 **VCONN1:** VCONN power to CC1 control
0: Disable
1: Enable

Bit 0 **PU1:** Pull up current to CC1 control
0: Disable
1: Enable

• **CC2_CONTROL Register**

Bit	7	6	5	4	3	2	1	0
Name	VBUSOK	—	—	—	PDWN2	TXE2	VCONN2	PU2
R/W	RO	—	—	—	RWH	RWH	RWH	RWH
POR	1	—	—	—	1	0	0	0

Bit 7 **VBUSOK:** VBUS active status
0: VBUS is not active
1: VBUS is active

Bit 6~4 Unimplemented, read as "0"

Bit 3 **PDWN2:** 5.1kΩ pull down resistor to CC2 control
0: Disable
1: Enable

Bit 2 **TXE2:** CC2 PD driver control
0: Disable
1: Enable

Bit 1 **VCONN2:** VCONN power to CC2 control
0: Disable
1: Enable

Bit 0 **PU2:** Pull up current to CC2 control
0: Disable
1: Enable

• **CC_SEL Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	VCONN_SWAP_OFF	VCONN_SWAP_ON	CC_SEL1	CC_SEL0
R/W	—	—	—	—	WC	WC	RWH	RWH
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3 **VCONN_SWAP_OFF:** VCONN via Type-C FSM control
0: Reserved
1: Turn off

Bit 2 **VCONN_SWAP_ON:** VCONN via Type-C FSM control
0: Reserved
1: Turn on

Bit 1~0 **CC_SEL1~CC_SEL0:** CC2 PD driver control
00: Reserved
01: CC1 select
10: CC2 select
11: Reserved

• **USB_C_STATUS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	TYPE_C_DET7	TYPE_C_DET6	CC_ORIENT5	CC_ORIENT4	TYPE_C_RSLT3	TYPE_C_RSLT2	TYPE_C_RSLT1	TYPE_C_RSLT0
R/W	RO	RO	RO	RO	RO	RO	RO	RO
POR	0	0	0	0	0	0	0	0

Bit 7~6 **TYPE_C_DET7~TYPE_C_DET6:** Type-C detection status
00: Type-C detection has not started
01: Type-C detection is ongoing
10: Type-C detection is completed (Type-C result may be read)
11: Reserved

Bit 5~4 **CC_ORIENT5~CC_ORIENT4:** CC1/CC2 connection status
00: No or unresolved connection is detected
01: Position 1 (Normal Orientation – CC1)
10: Position 2 (Normal Orientation – CC2)
11: A fault has occurred during detection

Bit 3~0 **TYPE_C_RSLT3~TYPE_C_RSLT0:** Type-C detection result status
0000: Nothing is attached
0001: SRC with Default current capability is attached
0010: SRC with 1.5A current capability is attached
0011: SRC with 3A current capability is attached
0100: SNK is attached
0101: Debug Accessory is attached
0110: Audio Accessory is attached
0111: Powered Accessory is attached
1000~1110: Reserved
1111: Undetermined

• USB_C_STATUS2 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	VBUS_REQ	PD_NOT_ALLOWED	—	—	OVRTEMP	SHORT
R/W	—	—	RO	RO	—	—	RO	RO
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **VBUS_REQ:** Indicates the state the VBUS switch should be on
0: VBUS is not required
1: VBUS is now required

Bit 4 **PD_NOT_ALLOWED:** PD allowed status
0: PD is allowed
1: PD is not allowed

Bit 3~2 Unimplemented, read as "0"

Bit 1 **OVRTEMP:** Over temperature status
0: Not over temperature
1: Over temperature

Bit 0 **SHORT:** Reflect short logic input
This indicates that the VCONN output voltage switch is on and there is an unsafe voltage across it. This signal will typically glitch when VCONN is turned on, and remain high during an overcurrent situation. A sustained short signal will cause an I_SHORT interrupt. Hence this bit is normally used only for debug.

• USB_C_STATUS3 Register

Bit	7	6	5	4	3	2	1	0
Name	TYPEC_ACTIVE	—	—	—	—	—	—	—
R/W	RO	—	—	—	—	—	—	—
POR	1	—	—	—	—	—	—	—

Bit 7 **TYPEC_ACTIVE:** FSM active status
0: FSM is note active
1: FSM is active

Bit 6~0 Unimplemented, read as "0"

• **CC1_CMP Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	DET_3A	DET_1P5A	DET_DEF	DET_RD	DET_RA
R/W	—	—	—	RO	RO	RO	RO	RO
POR	—	—	—	0	0	0	0	0

Bit 7~5 Unimplemented, read as "0"

Bit 4 **DET_3A:** CC1 3A detection status
0: 3A is not detected
1: 3A is detected

Bit 3 **DET_1P5A:** CC1 1.5A detection status
0: 1.5A is not detected
1: 1.5A is detected

Bit 2 **DET_DEF:** CC1 default current detection status
0: Default current is not detected
1: Default current is detected

Bit 1 **DET_RD:** CC1 R_D detection status
0: R_D is not detected
1: R_D is detected

Bit 0 **DET_RA:** CC1 R_A detection status
0: R_A is not detected
1: R_A is detected

• **CC2_CMP Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	DET_3A	DET_1P5A	DET_DEF	DET_RD	DET_RA
R/W	—	—	—	RO	RO	RO	RO	RO
POR	—	—	—	0	0	0	0	0

Bit 7~5 Unimplemented, read as "0"

Bit 4 **DET_3A:** CC2 3A detection status
0: 3A is not detected
1: 3A is detected

Bit 3 **DET_1P5A:** CC2 1.5A detection status
0: 1.5A is not detected
1: 1.5A is detected

Bit 2 **DET_DEF:** CC2 default current detection status
0: Default current is not detected
1: Default current is detected

Bit 1 **DET_RD:** CC2 R_D detection status
0: R_D is not detected
1: R_D is detected

Bit 0 **DET_RA:** CC2 R_A detection status
0: R_A is not detected
1: R_A is detected

• **CC1_STATUS Register**

Bit	7	6	5	4	3	2	1	0
Name	—	SRC_RX16	SRC_RX15	SRC_RP1	PWR3A_SNK1	PWR1P5A_SNK1	PWRDEF_SNK1	SNK_RP1
R/W	—	RO	RO	RO	RO	RO	RO	RO
POR	—	0	0	0	0	0	0	0

Bit 7 Unimplemented, read as "0"

Bit 6~5 **SRC_RX16~SRC_RX15:** CC1 SRC detection status
00: Nothing is detected after $t_{CCDebounce}$ (SRC.open)
01: R_D is detected after $t_{CCDebounce}$ (SRC.R_D)
10: R_A is detected after $t_{CCDebounce}$ (SRC.R_A)
11: Reserved

Bit 4 **SRC_RP1:** CC1 SRC detection status
0: Nothing is detected (CC1 above maximum V_{RD})
1: R_D or R_A pull-down is detected (CC1 below maximum V_{RD})

Bit 3 **PWR3A_SNK1:** CC1 SNK 3A detection status
0: Nothing is detected
1: SRC with 3A current capability is detected

Bit 2 **PWR1P5A_SNK1:** CC1 SNK 1.5A detection status
0: Nothing is detected
1: SRC with 1.5A current capability is detected

Bit 1 **PWRDEF_SNK1:** CC1 SNK default current detection status
0: Nothing is detected
1: SRC with default current capability is detected

Bit 0 **SNK_RP1:** CC1 SNK R_P detection status
0: Nothing is detected (CC1 below maximum V_{RA}) (SNK.open)
1: R_P pull-up is detected (CC1 above minimum V_{RD}) (SNK.R_P)

• **CC2_STATUS Register**

Bit	7	6	5	4	3	2	1	0
Name	—	SRC_RX26	SRC_RX25	SRC_RP2	PWR3A_SNK2	PWR3A_SNK2	PWRDEF_SNK2	SNK_RP2
R/W	—	RO	RO	RO	RO	RO	RO	RO
POR	—	0	0	0	0	0	0	0

Bit 7 Unimplemented, read as "0"

Bit 6~5 **SRC_RX26~SRC_RX25:** CC2 SRC detection status
00: Nothing is detected after $t_{CCDebounce}$ (SRC.open)
01: R_D is detected after $t_{CCDebounce}$ (SRC.R_D)
10: R_A is detected after $t_{CCDebounce}$ (SRC.R_A)
11: Reserved

Bit 4 **SRC_RP2:** CC2 SRC detection status
0: Nothing is detected (CC2 above maximum V_{RD})
1: R_D or R_A pull-down is detected (CC2 below maximum V_{RD})

Bit 3 **PWR3A_SNK2:** CC2 SNK 3A detection status
0: Nothing is detected
1: SRC with 3A current capability is detected

Bit 2 **PWR1P5A_SNK2:** CC2 SNK 1.5A detection status
0: Nothing is detected
1: SRC with 1.5A current capability is detected

Bit 1 **PWRDEF_SNK2:** CC2 SNK default current detection status
0: Nothing is detected
1: SRC with default current capability is detected

Bit 0 **SNK_RP2:** CC2 SNK R_P detection status
0: Nothing is detected (CC2 below maximum V_{RA}) (SNK.open)
1: R_P pull-up is detected (CC2 above minimum V_{RD}) (SNK.R_P)

• VBUS_MON Register

Bit	7	6	5	4	3	2	1	0
Name	VBUS_MON_EN	COMP	DAC5	DAC4	DAC3	DAC2	DAC1	DAC0
R/W	R/W	RO	RWH	RWH	RWH	RWH	RWH	RWH
POR	0	0	0	0	0	0	0	0

Bit 7 **VBUS_MON_EN:** VBUS monitor comparator control
0: Disable
1: Enable

Bit 6 **COMP:** VBUS higher than DAC set threshold
0: VBUS is not higher
1: VBUS is higher

Bit 5~0 **DAC5~DAC0:** Scaled VBUS threshold value
For 48V EPR configuration, a 6-bit DAC is used to detect voltage drop. With recommended external divider, the following method is used to set the VBUS voltage when voltage drop detection is required:
DAC=1.5×(VBUS-6)

• AFE_TRIM2 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	TRIM_CCDRV1	TRIM_CCDRV0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	1	0

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **TRIM_CCDRV1~TRIM_CCDRV0:** CC driver rise/fall trim
00: Slowest
01~10: Mid-point
11: Fastest

• AFE_TRIM3 Register

Bit	7	6	5	4	3	2	1	0
Name	—	TRIM_SLICE2	TRIM_SLICE1	TRIM_SLICE0	—	—	—	—
R/W	—	R/W	R/W	R/W	—	—	—	—
POR	—	0	0	0	—	—	—	—

Bit 7 Unimplemented, read as "0"

Bit 6~4 **TRIM_SLICE2~TRIM_SLICE0:** Trim for the slicer when CDR_SELECT=1
The valid values are 3'h4 : 3'h0, with 3'h2 being the mid position of the trim. It is likely that after characterization these bits can be fixed, and always set to the same value on boot. Use a default value of 3'h7.

Bit 3~0 Unimplemented, read as "0"

• **POWER Register**

Bit	7	6	5	4	3	2	1	0
Name	VCONN_CTRL_EXT	—	—	—	PWR3	PWR2	PWR1	PWR0
R/W	R/W	—	—	—	R/W	R/W	R/W	R/W
POR	0	—	—	—	1	1	1	1

Bit 7 **VCONN_CTRL_EXT:** External VCONN generator enable control
0: Use internal VCONN generator
1: Use external VCONN generator

Bit 6~4 Unimplemented, read as "0"

Bit 3~0 **PWR3~PWR0:** Power enable control
0000: All Disable
0001: AFE Power Enable
0010: Wake Power Enable
0100: Bandgap and LDO Power Enable
1000: Oscillator Enable
1101: AFE Power, Bandgap and LDO Power, Oscillator Enable
1111: AFE Power, Wake Power, Bandgap and LDO Power, Oscillator Enable
Others: Reserved

• **IRQ1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	I_VBUS_DROP	—	I_OVRTEMP	I_SHORT
R/W	—	—	—	—	RC	—	RC	RC
POR	—	—	—	—	0	—	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3 **I_VBUS_DROP:** Indicates VBUS has dropped below threshold set by DAC[5:0]
0: VBUS is not dropped below threshold
1: VBUS is dropped below threshold

Bit 2 Unimplemented, read as "0"

Bit 1 **I_OVRTEMP:** OTP interrupt
0: OTP is not interrupt
1: OTP is interrupt

Bit 0 **I_SHORT:** Short interrupt
0: Short is not interrupt
1: Short is interrupt

• **IRQ2 Register**

Bit	7	6	5	4	3	2	1	0
Name	I_CC_CHANGE	I_PD_RX	I_PD_HR_RX	I_PD_CR_RX	I_PD_TX_OK	I_PD_TX_FAIL	I_FAST_SWAP	I_TX_DISCARD
R/W	RC	RC	RC	RC	RC	RC	RC	RC
POR	1	0	0	0	0	0	0	0

Bit 7 **I_CC_CHANGE:** Type-C status changed
0: Type-C status is not changed
1: Type-C status is changed

Bit 6 **I_PD_RX:** PD-message received
0: PD-message is not received
1: PD-message is received

Bit 5 **I_PD_HR_RX:** PD-HardReset received
0: PD-HardReset is not received
1: PD-HardReset is received

Bit 4 **I_PD_CR_RX:** PD-CableReset received
0: PD-CableReset is not received
1: PD-CableReset is received

Bit 3 **I_PD_TX_OK:** PD-Transmit success
0: PD-Transmit is not success
1: PD-Transmit is success

Bit 2 **I_PD_TX_FAIL:** PD-Transmit failure
0: PD-Transmit is not failure
1: PD-Transmit is failure

Bit 1 **I_FAST_SWAP:** Fast Role Swap occurred
0: Fast Role Swap is not occurred
1: Fast Role Swap is occurred

Bit 0 **I_TX_DISCARD:** PD transmit discarded due to incoming message
0: PD transmit was not discarded
1: PD transmit was discarded
Refer to the TX_RESULT[2:0] bits in the PD_STATUS register.

- **IRQ_MSK1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	M_VBUS_DROP	—	M_OVRTEMP	M_SHORT
R/W	—	—	—	—	R/W	—	R/W	R/W
POR	—	—	—	—	0	—	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3 **M_VBUS_DROP:** VBUS_DROP interrupt control
0: Disable
1: Enable

Bit 2 Unimplemented, read as "0"

Bit 1 **M_OVRTEMP:** OTP interrupt control
0: Disable
1: Enable

Bit 0 **M_SHORT:** Short interrupt control
0: Disable
1: Enable

- **IRQ_MSK2 Register**

Bit	7	6	5	4	3	2	1	0
Name	M_CC_CHANGE	M_PD_RX	M_PD_HR_RX	M_PD_CR_RX	M_PD_TX_OK	M_PD_TX_FAIL	M_FAST_SWAP	—
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	—
POR	0	0	0	0	0	0	0	—

Bit 7 **M_CC_CHANGE:** Type-C status changed interrupt control
0: Disable
1: Enable

Bit 6 **M_PD_RX:** PD-message received interrupt control
0: Disable
1: Enable

Bit 5 **M_PD_CR_RX:** PD-HardReset received interrupt control
0: Disable
1: Enable

Bit 4 **M_PD_TX_OK:** PD-CableReset received interrupt control
0: Disable
1: Enable

Bit 3 **M_VBUS_DROP:** PD-Transmit success interrupt control
0: Disable
1: Enable

Bit 2 **M_PD_TX_FAIL:** PD-Transmit failure interrupt control
0: Disable
1: Enable

Bit 1 **M_FAST_SWAP:** Fast Role Swap interrupt control
0: Disable
1: Enable

Bit 0 Unimplemented, read as "0"

- **PD_CFG1 Register**

Bit	7	6	5	4	3	2	1	0
Name	ID_IN-SERT	VBUS_ HIGH_VOLT	GUIDE_ TRY_SNK	—	RESET_ MSG_ID	SOP_TO_ RESET2	SOP_TO_ RESET1	SOP_TO_ RESET0
R/W	R/W	R/W	R/W	—	WC	R/W	R/W	R/W
POR	0	0	0	—	0	0	0	0

Bit 7 **ID_INSERT:** PD-FSM status control
0: PD-FSM will not change outgoing message – In this case software is responsible for providing the ID bits in the header
1: PD-FSM will keep track of Transmitted IDs for all SOP* and insert the correct value into the header at bit 11 ~ bit 9, no other bits are touched

Bit 6 **VBUS_HIGH_VOLT:** PD-FSM BIST Mode 2 control
0: PD-FSM will respond to request for BIST Mode 2 for compliance
1: PD-FSM will not respond to request for BIST Mode 2 for compliance

Note: The software must keep this bit updated according to VBUS voltage. Set to 1 if VBUS is above vSafe5V. This is required by the specification. Note that the original purpose of this was to prevent malicious use of BIST to force a port to high voltage which continues even after disconnect. However, the specification now limits the BIST length to 60ms.

Bit 5 **GUIDE_TRY_SNK:** Use the TRY_SINK modified Type-C FSM
0: Not use
1: Use

Bit 4 Unimplemented, read as "0"

Bit 3 **RESET_MSG_ID:** Write 1 to clear the MessageID for SOP_TO_RESET type messages control
0: Reserved
1: Clear the MessageID

Bit 2~0 **SOP_TO_RESET2~SOP_TO_RESET0:** Which SOP to reset MessageID for. Typically used to reset ID during a swap operation
001: SOP
010: SOP'
010: SOP''
100: Debug_SOP
101: Debug_SOP"
110~111: Reserved

• **PD_CFG2 Register**

Bit	7	6	5	4	3	2	1	0
Name	FAST_ SWAP_SNK	FAST_ SWAP_SRC	CDR_ SELECT	SOP_ RCV4	SOP_ RCV3	SOP_ RCV2	SOP_ RCV1	SOP_ RCV0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **FAST_SWAP_SNK:** Fast Role Swap sink function enable control
0: Disable
1: Enable

Bit 6 **FAST_SWAP_SRC:** Fast Role Swap source function enable control
0: Disable
1: Enable

Bit 5 **CDR_SELECT:** Select between versions of the CC CDR
0: 3-level slicer
1: AC coupled
Note that this bit also controls rxMode signals to the AFE.

Bit 4~0 **SOP_RCV4~SOP_RCV0:** Select which SOP to receive
00000: No SOP is received
xxxx1: SOP is received
xxx1x: SOP' is received
xx1xx: SOP'' is received
x1xxx: SOP_debug' is received
1xxxx: SOP_debug'' is received
Note that multiple SOP types can be received at the same time.

• **PD_CFG3 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	P_DATA_ ROLE_DP	P_PWR_ ROLE_DP	P_DATA_ ROLE_PR	P_PWR_ ROLE_PR	P_DATA_ ROLE_SOP	P_PWR_ ROLE_SOP
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **P_DATA_ROLE_DP:** SOP''. Current Port Data Role. Transmit header bit 5
0: Bit is cleared to 0
1: Bit is set to 1
Cable communication to the remote end of the cable (future use).

Bit 4 **P_PWR_ROLE_DP:** SOP''. Current Port Power Role. Transmit header bit 8
Cable communication to the remote end of the cable (future use).
0: Bit is cleared to 0
1: Bit is set to 1

Bit 3 **P_DATA_ROLE_PR:** SOP'. Current Port Data Role. Transmit header bit 5
0: Bit is cleared to 0
1: Bit is set to 1
Cable communication to the near end of the cable.

Bit 2 **P_PWR_ROLE_PR:** SOP'. Current Port Power Role. Transmit header bit 8
0: Bit is cleared to 0
1: Bit is set to 1
Cable communication to the near end of the cable.

Bit 1 **P_DATA_ROLE_SOP:** SOP. Current Port Data Role. Transmit header bit 5
0: Bit is cleared to 0
1: Bit is set to 1

Bit 0 **P_PWR_ROLE_SOP:** SOP. Current Port Power Role. Transmit header bit 8
0: Bit is cleared to 0
1: Bit is set to 1

• **SHORT_PROTECT Register**

Bit	7	6	5	4	3	2	1	0
Name	SHORT_RESET	SHORT_TIME6	SHORT_TIME5	SHORT_TIME4	SHORT_TIME3	SHORT_TIME2	SHORT_TIME1	SHORT_TIME0
R/W	WC	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	1	0	1	0	0	1

Bit 7~6 **SHORT_RESET:** Short reset control
0: Reserved
1: Reset
Note: 1. Write 1 to reset VCONN power after maximum short time has been reached.
2. To prevent over-temperature, protect should not be reset more than once per second.

Bit 6~0 **SHORT_TIME6~SHORT_TIME0:** Maximum short time before VCONN is shut off – One LSB is 0.488ms

• **PD_STATUS Register**

Bit	7	6	5	4	3	2	1	0
Name	FAST_SWAP	—	RX_RESULT5	RX_RESULT4	RX_RESULT3	TX_RESULT2	TX_RESULT1	TX_RESULT0
R/W	RO	—	RO	RO	RO	RO	RO	RO
POR	0	—	0	0	0	0	0	0

Bit 7 **FAST_SWAP:** Indicates Fast Role Swap has happened
0: Fast Role Swap has not happened
1: Fast Role Swap has happened

Bit 6 Unimplemented, read as "0"

Bit 5~3 **RX_RESULT5~RX_RESULT3:** Received status
000: No operation
001: Message Received - Read buffer
010: Hard reset from remote
011: Cable reset from remote
100~111: Reserved

Bit 2~0 **TX_RESULT2~TX_RESULT0:** Transmitter status
000: No operation
001: Message sent with success
010: Transmission error
011: Transmitter busy
100: Transmit discarded due to incoming message
101~111: Reserved

• **RX_STATUS Register**

Bit	7	6	5	4	3	2	1	0
Name	RX_DATA	RX_OVERRUN	—	—	—	—	—	RX_CLEAR
R/W	RO	RO	—	—	—	—	—	WC
POR	0	0	—	—	—	—	—	0

Bit 7 **RX_DATA:** Indicates buffer data is available
0: Buffer data is not available
1: Buffer data is available

Bit 6 **RX_OVERRUN:** New data has arrived without the previous data having been read
0: Read
1: Not read
Cleared by writing a "1" to RX_CLEAR.

Bit 5~1 Unimplemented, read as "0"

Bit 0 **RX_CLEAR:** Write a "1" to this bit to indicate data has been read
0: Reserved
1: Indicates data has been read
If there is another packet in buffer, RX_DATA will remain 1.

• **RX_INFO Register**

Bit	7	6	5	4	3	2	1	0
Name	RX_ BYTES7	RX_ BYTES6	RX_ BYTES5	RX_ BYTES4	RX_ BYTES3	RX_ SOP2	RX_ SOP1	RX_ SOP0
R/W	RO	RO	RO	RO	RO	RO	RO	RO
POR	0	0	0	0	0	0	0	0

Bit 7~3 **RX_BYTES7~RX_BYTES3:** Actual number of bytes received

Bit 2~0 **RX_SOP2~RX_SOP0:** SOP type of the received message
000: No Message
001: SOP
010: SOP'
011: SOP''
100: Debug SOP'
101: Debug SOP''
110~111: Reserved

• **TX_COMMAND Register**

Bit	7	6	5	4	3	2	1	0
Name	TX_ CMD7	TX_ CMD6	TX_ CMD5	—	—	TX_ WAIT_RP	TX_ START	TXBUF_ READY
R/W	R/W	R/W	R/W	—	—	R/W	WC	RO
POR	0	0	0	—	—	0	0	1

Bit 7~5 **TX_CMD7~TX_CMD5:** Transmit Commands control
000: NOP
001: Send the message in buffer
010: Send CableReset
011: Send HardReset
100: Start BIST Mode 2 for 45ms
101~111: Unused

Bit 4~3 Unimplemented, read as "0"

Bit 2 **TX_WAIT_RP:** TX Wait R_P control
0: TX does not wait
1: TX will hold until R_P=3A

Bit 1 **TX_START:** A write to this bit starts transmission of what is in the TX buffer
0: Not transmission
1: Starts transmission

Bit 0 **TXBUF_READY:** Transmission status
0: Transmission ongoing
1: Transmission complete
The software should check whether this bit is 1 before filling the transmit buffer.

• **TX_INFO Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	TX_RE-TRIES5	TX_RE-TRIES4	TX_RE-TRIES3	TX_ SOP2	TX_ SOP1	TX_ SOP0
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	1	1	0	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5~3 **TX_RETRIES5~TX_RETRIES3:** Number of retries to be attempted for this message

Bit 2~0 **TX_SOP2~TX_SOP0:** SOP type of the transmitted message

000: Not Valid
001: SOP
010: SOP'
011: SOP''
100: Debug SOP'
101: Debug SOP''
110~111: Reserved

• **RX_PACKET_DATAm Register (m=61~90)**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0:** Receive packet data

The SOP and EOP symbols are not included in the receive data. The first byte of the data is the 8 LSB's of the packet header, and is ordered consistently with the power delivery specification. Also:

- The GoodCRC packets are consumed by the logic and do not appear in the receive buffer.
- The packet's CRC is not included in the byte count. However, when space in the buffer is available, the CRC field appears after the data in the buffer.

• **TX_PACKET_DATAn Register (n=91~120)**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0:** Transmit packet data

The SOP and EOP symbols are not included in the transmit data. The first byte of the data is the 8 LSB's of the packet header, and is ordered consistently with the power delivery specification.

• **C_OVP Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	ENP2	ENP1	STP2	STP1
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **ENP2~ENP1:** Associated Protect blocks enable control
00: Disable
01: CC1 Protect blocks enable
10: CC2 Protect blocks enable
11: CC1 and CC2 Protect blocks enable
The Protect blocks must be enabled before they are started.

Bit 1~0 **STP2~STP1:** Protect blocks for the associated CC1/CC2 Start control
00: Disable
01: CC1 OVP Start enable
10: CC2 OVP Start enable
11: CC1 and CC2 OVP Start enable
These can be started at boot time, or to save power only started when needed. The protect blocks must be first enabled with CC1/CC2.

PD PHY Reset

When the system is powered on for the first time, the SYSRESETN line needs to be sent a signal from low to high with a width of 30μs to complete the reset action. After this, send the I²C command to set the relevant registers.

SVBUS External Divider Resistor Configuration

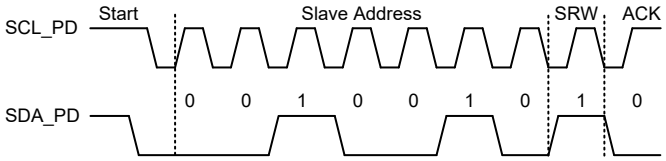
Component	Connection	Note
220kΩ + 680Ω 1% resistors	VBUS to SVBUS pin	—
12kΩ + 1kΩ 1% resistors	SVBUS pin to ground	Good analog ground connection
10nF capacitor	SVBUS pin to ground	Good analog ground connection

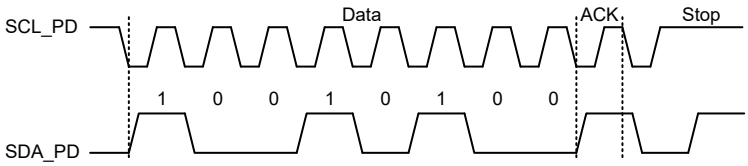
External Divider for 48V EPR Scaled VBUS (SVBUS Input)

PD PHY I²C Function

The I²C serial interface is a two line interface, a serial data line, SDA_PD, and serial clock line, SCL_PD. As many devices may be connected together on the same bus, their outputs are both open drain types. For this reason it is necessary that external pull-high resistors are connected to these outputs. Note that no chip select line exists, as each device on the I²C bus is identified by a unique address which will be transmitted and received on the I²C bus. When two devices communicate with each other on the bidirectional I²C bus, one is known as the master device and one as the slave device. Both master and slave can transmit and receive data, however, it is the master device that has overall control of the bus. For the device, which only operates in slave mode, there are two methods of transferring data on the I²C bus, the slave transmit mode and the slave receive mode.

Communication on the I²C bus requires four separate steps, a START signal, a slave device address transmission, a data transmission and finally a STOP signal. When a START signal is placed on the I²C bus, all devices on the bus will receive this signal and be notified of the imminent arrival of data on the bus. The first seven bits of the data will be the slave address, the 8th bit is the read/write bit whose value will be placed in the SRW bit. This bit will be checked by the slave device to determine whether to go into transmit or receive mode. The transmitted data is 8-bit wide and is transmitted after the slave device has acknowledged receipt of its slave address. The order of serial bit transmission is the MSB first and the LSB last. After receipt of 8 bits of data, the receiver must transmit an acknowledge signal, level “0”, before it can receive the next data byte. If the slave transmitter does not receive an acknowledge bit signal from the master receiver, then the slave transmitter will release the SDA_PD line to allow the master to send a STOP signal to release the I²C Bus.



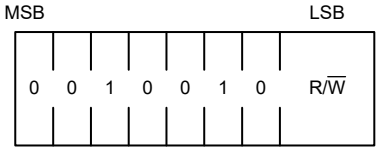


S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (1 bit)
P=Stop (1 bit)

S	SA	SR	M	D	A	D	A	.....	S	SA	SR	M	D	A	D	A	.....	P

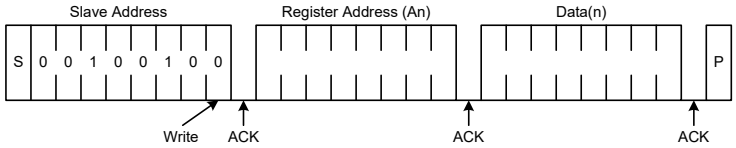
Device Addressing

The slave address byte is the first byte received following the START condition from the master device. The first seven bits of the first byte make up the slave address. The eighth bit defines whether a read or write operation is to be performed. When the R/$\overline{W}$ bit is "1", then a read operation is selected. A "0" selects a write operation. The device address bits are "0010010". When an address byte is sent, the device compares the first seven bits after the START condition. If they match, the device outputs an acknowledge on the SDA_PD line.

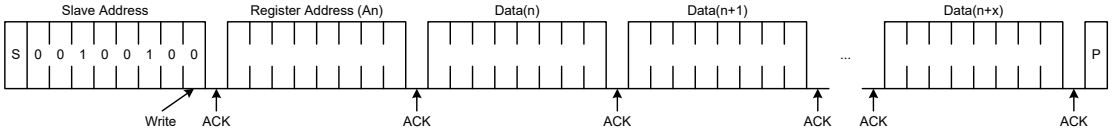


The first byte after the START

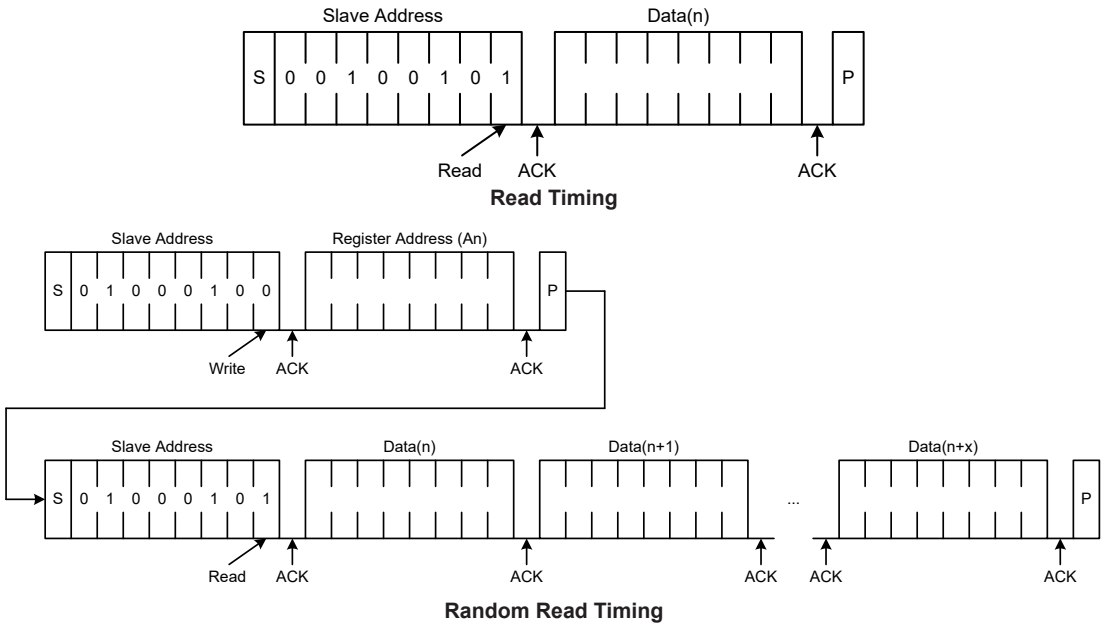
Write/Read Timing



Byte Write Timing



Page Write Timing



Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer Module or an A/D converter requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs. The device contains several external interrupt and internal interrupt functions. The external interrupts are generated by the action of the external INT0~INT1 pins, while the internal interrupts are generated by various internal functions such as the TMs, Time Bases, LVD, EEPROM, SIM, UART and the A/D converter, etc.

Interrupt Registers

Overall interrupt control, which basically means the setting of request flags when certain microcontroller conditions occur and the setting of interrupt enable bits by the application program, is controlled by a series of registers, located in the Special Purpose Data Memory, as shown in the accompanying table. The registers fall into three categories. The first is the INTC0~INTC3 registers which setup the primary interrupts, the second is the MFI0~MFI2 registers which setup the Multi-function interrupts. Finally there is an INTEG register to setup the external interrupt trigger edge type.

Each register contains a number of enable bits to enable or disable individual interrupts as well as interrupt flags to indicate the presence of an interrupt request. The naming convention of these follows a specific pattern. First is listed an abbreviated interrupt type, then the (optional) number of that interrupt followed by either an “E” for enable/disable bit or “F” for request flag.

Function	Enable Bit	Request Flag	Notes
Global	EMI	—	—
I^2C master mode	IICME	IICMF	—
UART	URE	URF	—
INTn Pin	INTnE	INTnF	n=0~1
INT2 Line (for PD PHY)	INT2E	INT2F	—
A/D Converter	ADE	ADF	—
Multi-function	MFnE	MFnF	n=0~2

Function	Enable Bit	Request Flag	Notes
Time Base	TBnE	TBnF	n=0~1
SIM	SIME	SIMF	—
EEPROM erase or write operation	DEE	DEF	—
LVD	LVE	LVF	—
PTM	PTMnPE	PTMnPF	n=0~3
	PTMnAE	PTMnAF	
STM	STMPE	STMPF	—
	STMAE	STMAF	

Interrupt Register Bit Naming Conventions

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	INT3S1	INT3S0	INT2S1	INT2S0	INT1S1	INT1S0	INT0S1	INT0S0
INTC0	—	INT0F	URF	IICMF	INT0E	URE	IICME	EMI
INTC1	MF1F	MF0F	ADF	INT1F	MF1E	MF0E	ADE	INT1E
INTC2	SIMF	TB1F	TB0F	MF2F	SIME	TB1E	TB0E	MF2E
INTC3	INT3F	INT2F	LVF	DEF	INT3E	INT2E	LVE	DEE
MFI0	PTM2AF	PTM2PF	PTM0AF	PTM0PF	PTM2AE	PTM2PE	PTM0AE	PTM0PE
MFI1	PTM3AF	PTM3PF	PTM1AF	PTM1PF	PTM3AE	PTM3PE	PTM1AE	PTM1PE
MFI2	—	—	STMAF	STMPF	—	—	STMAE	STMPE

Interrupt Register List

• INTEG Register

Bit	7	6	5	4	3	2	1	0
Name	INT3S1	INT3S0	INT2S1	INT2S0	INT1S1	INT1S0	INT0S1	INT0S0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **INT3S1~INT3S0:** Reserved
Note that these bits must be kept unchanged after power on.

Bit 5~4 **INT2S1~INT2S0:** Interrupt edge control for INT2 line
Note that the INT2 line is internally connected to the PD PHY interrupt output signal, BUS_INT, therefore the PD PHY interrupt uses the INT2 interrupt vector. These bits should be set to "01" after power on.

Bit 3~2 **INT1S1~INT1S0:** Interrupt edge control for INT1 pin
00: Disable
01: Rising edge
10: Falling edge
11: Rising and falling edges

Bit 1~0 **INT0S1~INT0S0:** Interrupt edge control for INT0 pin
00: Disable
01: Rising edge
10: Falling edge
11: Rising and falling edges

• **INTC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	INT0F	URF	IICMF	INT0E	URE	IICME	EMI
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7 Unimplemented, read as "0"

Bit 6 **INT0F:** INT0 interrupt request flag
0: No request
1: Interrupt request

Bit 5 **URF:** UART transfer interrupt request flag
0: No request
1: Interrupt request

Bit 4 **IICMF:** I^2C master mode interrupt request flag
0: No request
1: Interrupt request

Bit 3 **INT0E:** INT0 interrupt control
0: Disable
1: Enable

Bit 2 **URE:** UART transfer interrupt control
0: Disable
1: Enable

Bit 1 **IICME:** I^2C master mode interrupt control
0: Disable
1: Enable

Bit 0 **EMI:** Global interrupt control
0: Disable
1: Enable

• **INTC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	MF1F	MF0F	ADF	INT1F	MF1E	MF0E	ADE	INT1E
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **MF1F:** Multi-function 1 interrupt request flag
0: No request
1: Interrupt request

Bit 6 **MF0F:** Multi-function 0 interrupt request flag
0: No request
1: Interrupt request

Bit 5 **ADF:** A/D Converter interrupt request flag
0: No request
1: Interrupt request

Bit 4 **INT1F:** INT1 interrupt request flag
0: No request
1: Interrupt request

Bit 3 **MF1E:** Multi-function 1 interrupt control
0: Disable
1: Enable

Bit 2 **MF0E:** Multi-function 0 interrupt control
0: Disable
1: Enable

Bit 1 **ADE:** A/D Converter interrupt control
0: Disable
1: Enable

Bit 0 **INT1E:** INT1 interrupt control
0: Disable
1: Enable

• **INTC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	SIMF	TB1F	TB0F	MF2F	SIME	TB1E	TB0E	MF2E
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **SIMF:** SIM Interrupt request flag
0: No request
1: Interrupt request

Bit 6 **TB1F:** Time Base 1 interrupt request flag
0: No request
1: Interrupt request

Bit 5 **TB0F:** Time Base 0 interrupt request flag
0: No request
1: Interrupt request

Bit 4 **MF2F:** Multi-function 2 interrupt request flag
0: No request
1: Interrupt request

Bit 3 **SIME:** SIM Interrupt control
0: Disable
1: Enable

Bit 2 **TB1E:** Time Base 1 interrupt control
0: Disable
1: Enable

Bit 1 **TB0E:** Time Base 0 interrupt control
0: Disable
1: Enable

Bit 0 **MF2E:** Multi-function 2 interrupt control
0: Disable
1: Enable

• **INTC3 Register**

Bit	7	6	5	4	3	2	1	0
Name	INT3F	INT2F	LVF	DEF	INT3E	INT2E	LVE	DEE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **INT3F:** Reserved
Note that these bits must be kept unchanged after power on.

Bit 6 **INT2F:** INT2 interrupt request flag
0: No request
1: Interrupt request
Note that the INT2 line is internally connected to the PD PHY interrupt output signal, BUS_INT, therefore the PD PHY interrupt uses the INT2 interrupt vector.

Bit 5 **LVF:** LVD Interrupt request flag
0: No request
1: Interrupt request

Bit 4 **DEF:** Data EEPROM Interrupt request flag
0: No request
1: Interrupt request

Bit 3 **INT3E:** Reserved
Note that these bits must be kept unchanged after power on.

Bit 2 **INT2E:** INT2 interrupt control
0: Disable
1: Enable
Note that the INT2 line is internally connected to the PD PHY interrupt output signal, BUS_INT, therefore the PD PHY interrupt uses the INT2 interrupt vector.

Bit 1 **LVE:** LVD Interrupt control
0: Disable
1: Enable

Bit 0 **DEE:** Data EEPROM Interrupt control
0: Disable
1: Enable

• MFI0 Register

Bit	7	6	5	4	3	2	1	0
Name	PTM2AF	PTM2PF	PTM0AF	PTM0PF	PTM2AE	PTM2PE	PTM0AE	PTM0PE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **PTM2AF:** PTM2 Comparator A match Interrupt request flag
0: No request
1: Interrupt request

Bit 6 **PTM2PF:** PTM2 Comparator P match Interrupt request flag
0: No request
1: Interrupt request

Bit 5 **PTM0AF:** PTM0 Comparator A match Interrupt request flag
0: No request
1: Interrupt request

Bit 4 **PTM0PF:** PTM0 Comparator P match Interrupt request flag
0: No request
1: Interrupt request

Bit 3 **PTM2AE:** PTM2 Comparator A match Interrupt control
0: Disable
1: Enable

Bit 2 **PTM2PE:** PTM2 Comparator P match Interrupt control
0: Disable
1: Enable

Bit 1 **PTM0AE:** PTM0 Comparator A match Interrupt control
0: Disable
1: Enable

Bit 0 **PTM0PE:** PTM0 Comparator P match Interrupt control
0: Disable
1: Enable

• **MFI1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PTM3AF	PTM3PF	PTM1AF	PTM1PF	PTM3AE	PTM3PE	PTM1AE	PTM1PE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **PTM3AF:** PTM3 Comparator A match Interrupt request flag
0: No request
1: Interrupt request

Bit 6 **PTM3PF:** PTM3 Comparator P match Interrupt request flag
0: No request
1: Interrupt request

Bit 5 **PTM1AF:** PTM1 Comparator A match Interrupt request flag
0: No request
1: Interrupt request

Bit 4 **PTM1PF:** PTM1 Comparator P match Interrupt request flag
0: No request
1: Interrupt request

Bit 3 **PTM3AE:** PTM3 Comparator A match Interrupt control
0: Disable
1: Enable

Bit 2 **PTM3PE:** PTM3 Comparator P match Interrupt control
0: Disable
1: Enable

Bit 1 **PTM1AE:** PTM1 Comparator A match Interrupt control
0: Disable
1: Enable

Bit 0 **PTM1PE:** PTM1 Comparator P match Interrupt control
0: Disable
1: Enable

• **MFI2 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	STMAF	STMPF	—	—	STMAE	STMPE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **STMAF:** STM Comparator A match Interrupt request flag
0: No request
1: Interrupt request

Bit 4 **STMPF:** STM Comparator P match Interrupt request flag
0: No request
1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

Bit 1 **STMAE:** STM Comparator A match Interrupt control
0: Disable
1: Enable

Bit 0 **STMPE:** STM Comparator P match Interrupt control
0: Disable
1: Enable

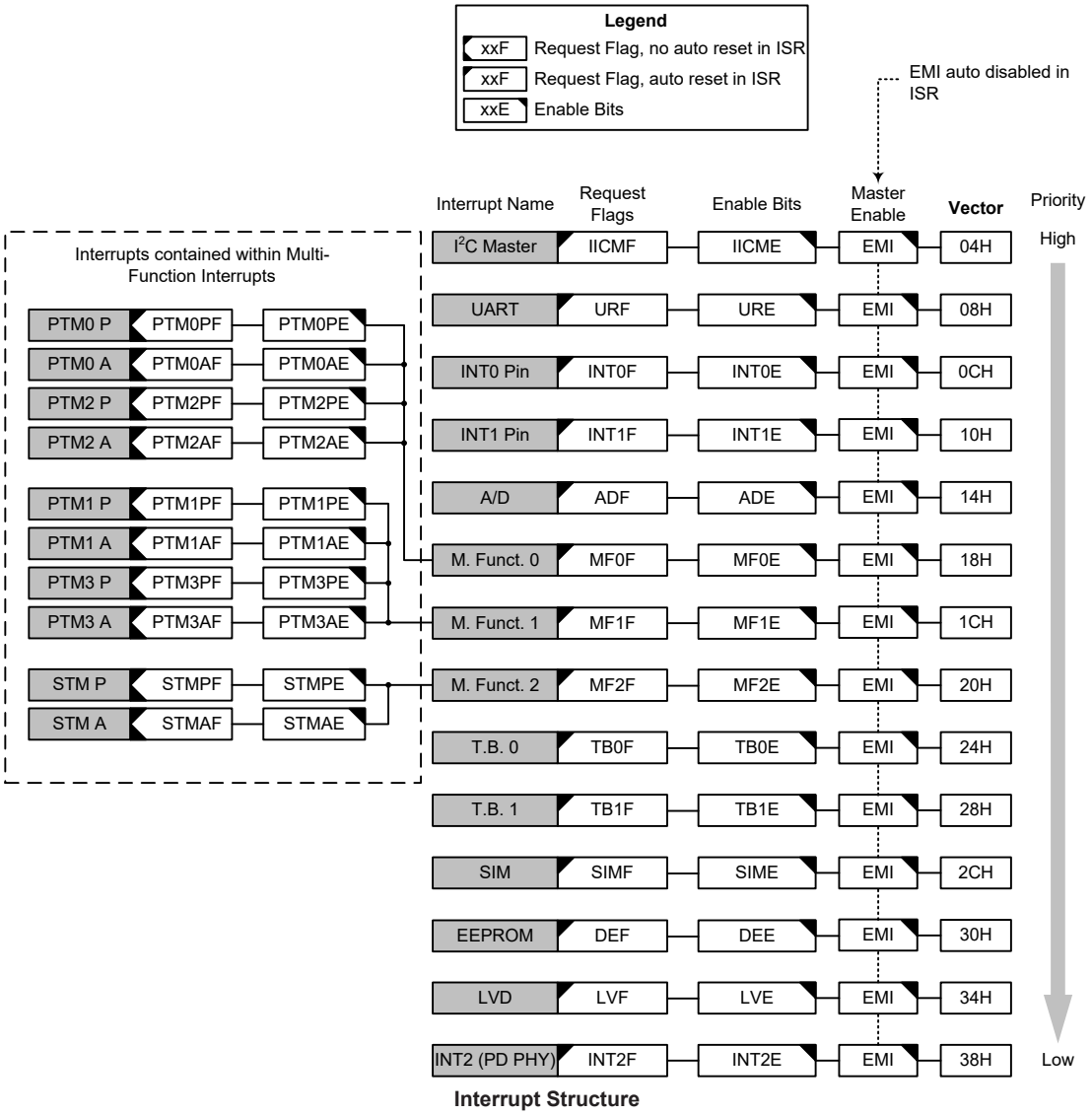
Interrupt Operation

When the conditions for an interrupt event occur, such as a TM Comparator P or Comparator A or A/D conversion completion, etc., the relevant interrupt request flag will be set. Whether the request flag actually generates a program jump to the relevant interrupt vector is determined by the condition of the interrupt enable bit. If the enable bit is set high then the program will jump to its relevant vector; if the enable bit is zero then although the interrupt request flag is set an actual interrupt will not be generated and the program will not jump to the relevant interrupt vector. The global interrupt enable bit, if cleared to zero, will disable all interrupts.

When an interrupt is generated, the Program Counter, which stores the address of the next instruction to be executed, will be transferred onto the stack. The Program Counter will then be loaded with a new address which will be the value of the corresponding interrupt vector. The microcontroller will then fetch their next instruction from this interrupt vector. The instruction at this vector will usually be a JMP which will jump to another section of program which is known as the interrupt service routine. Here is located the code to control the appropriate interrupt. The interrupt service routine must be terminated with a RETI, which retrieves the original Program Counter address from the stack and allows the microcontroller to continue with normal execution at the point where the interrupt occurred.

The various interrupt enable bits, together with their associated request flags, are shown in the accompanying diagrams with their order of priority. Some interrupt sources have their own individual vector while others share the same multi-function interrupt vector. Once an interrupt subroutine is serviced, all other interrupts will be blocked, as the global interrupt enable bit, EMI bit will be cleared automatically. This will prevent any further interrupt nesting from occurring. However, if other interrupt requests occur during this interval, although the interrupt will not be immediately serviced, the request flag will still be recorded.

If an interrupt requires immediate servicing while the program is already in another interrupt service routine, the EMI bit should be set after entering the routine to allow interrupt nesting. If the stack is full, the interrupt request will not be acknowledged, even if the related interrupt is enabled, until the Stack Pointer is decremented. If immediate service is desired, the stack must be prevented from becoming full. In case of simultaneous requests, the accompanying diagram shows the priority that is applied. All of the interrupt request flags when set will wake up the device if it is in SLEEP or IDLE Mode, however to prevent a wake-up from occurring the corresponding flag should be set before the device is in SLEEP or IDLE Mode.



Interrupt Structure

I²C Master Mode Interrupt

An I²C Master Mode Interrupt request will take place when the I²C Master Mode Interrupt request flag, IICMF, is set, which occurs when one of 9-bit data transfer/receive completion, STOP condition completion, START condition completion or address frame transmit completion situations happens. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and I²C Master Mode Interrupt enable bit, IICME, must first be set. When the interrupt is enabled, the stack is not full and any of the above described situations occurs, a subroutine call to the corresponding I²C Master Mode Interrupt vector, will take place. When the interrupt is serviced, the IICMF flag will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

UART Interrupt

The UART Interrupt is controlled by several UART transfer conditions. When one of these conditions occurs, an interrupt pulse will be generated to get the attention of the microcontroller. These conditions are a transmitter data register empty, transmitter idle, receiver reaching FIFO trigger level, receiver overrun, address detect and an RX/TX pin wake-up. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and UART Interrupt enable bit, URE, must first be set. When the interrupt is enabled, the stack is not full and any of the conditions described above occurs, a subroutine call to the corresponding UART Interrupt vector, will take place. When the interrupt is serviced, the UART Interrupt flag, URF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts. However, the USR register flags will only be cleared when certain actions are taken by the UART, the details of which are given in the UART section.

External Interrupts

The external interrupts are controlled by signal transitions on the pins INT0~INT1. An external interrupt request will take place when the external interrupt request flags, INT0F~INT1F, are set, which will occur when a transition, whose type is chosen by the edge select bits, appears on the external interrupt pins. To allow the program to branch to their respective interrupt vector address, the global interrupt enable bit, EMI, and respective external interrupt enable bits, INT0E~INT1E, must first be set. Additionally the correct interrupt edge type must be selected using the INTEG register to enable the external interrupt function and to choose the trigger edge type. As the external interrupt pins are pin-shared with I/O pins, they can only be configured as external interrupt pins if their external interrupt enable bit in the corresponding interrupt register has been set and the external interrupt pin is selected by the corresponding pin-shared function selection bits. The pin must also be setup as an input by setting the corresponding bit in the port control register. When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pins, a subroutine call to the corresponding external interrupt vector, will take place. When the interrupt is serviced, the external interrupt request flags, INT0F~INT1F, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that any pull-high resistor selections on the external interrupt pins will remain valid even if the pin is used as an external interrupt input.

The INTEG register is used to select the type of active edge that will trigger the external interrupt. A choice of either rising or falling or both edge types can be chosen to trigger an external interrupt. Note that the INTEG register can also be used to disable the external interrupt function.

PD PHY Interrupt

The PD PHY interrupt uses the INT2 interrupt vector as the INT2 line is internally connected to the PD PHY interrupt output signal, BUS_INT.

When a Type-C identification is completed, an interrupt signal BUS_INT will be generated to get the attention of the microcontroller. At this point a rising edge transition appears on the INT2 line, the INT2 interrupt request flag, INT2F, is set and an INT2 interrupt request will take place. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and respective INT2 interrupt enable bit, INT2E, must first be set. When the interrupt is enabled, the stack is not full and the correct transition type appears on the INT2 signal, a subroutine call to the INT2 interrupt vector will take place. When the interrupt is serviced, the INT2 interrupt request flag, INT2F, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

In addition, the correct interrupt edge type is rising edge, therefore the INTEG register should be set to “01” to trigger the INT2 interrupt.

A/D Converter Interrupt

The A/D Converter Interrupt is controlled by the termination of an A/D conversion process. An A/D Converter Interrupt request will take place when the A/D Converter Interrupt request flag, ADF, is set, which occurs when the A/D conversion process finishes. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and A/D Interrupt enable bit, ADE, must first be set. When the interrupt is enabled, the stack is not full and the A/D conversion process has ended, a subroutine call to the corresponding A/D Converter Interrupt vector, will take place. When the interrupt is serviced, the ADF flag will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Multi-function Interrupts

Within the device there are three Multi-function interrupts. Unlike the other independent interrupts, these interrupts have no independent source, but rather are formed from other existing interrupt sources, namely the TM interrupts.

A Multi-function interrupt request will take place when any of the Multi-function interrupt request flags MFnF are set. The Multi-function interrupt flags will be set when any of their included functions generate an interrupt request flag. When the Multi-function interrupt is enabled and the stack is not full, and either one of the interrupts contained within each of Multi-function interrupt occurs, a subroutine call to one of the Multi-function interrupt vectors will take place. When the interrupt is serviced, the related Multi-Function request flag will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

However, it must be noted that, although the Multi-function Interrupt request flags will be automatically reset when the interrupt is serviced, the request flags from the original source of the Multi-function interrupts will not be automatically reset and must be manually reset by the application program.

TM Interrupts

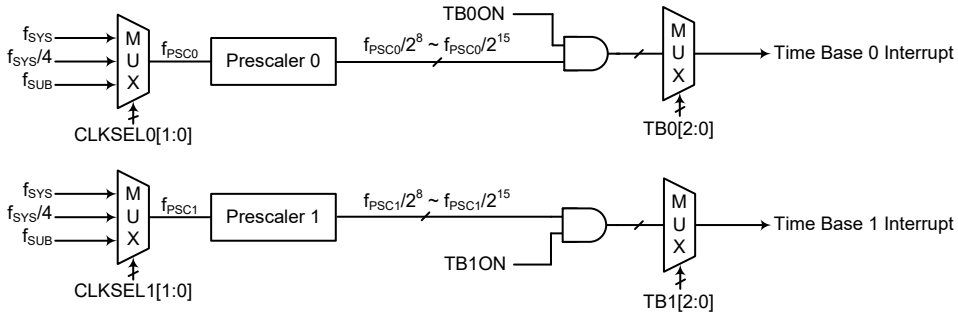
The Standard and Periodic TMs each have two interrupts, one comes from the comparator A match situation and the other comes from the comparator P match situation. All of the TM interrupts are contained within the Multi-function Interrupts. For all of the TM types there are two interrupt request flags and two enable control bits. A TM interrupt request will take place when any of the TM request flags are set, a situation which occurs when a TM comparator P or A match situation happens.

To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, respective TM Interrupt enable bit, and relevant Multi-function Interrupt enable bit, MFnE, must first be set. When the interrupt is enabled, the stack is not full and a TM comparator match situation occurs, a subroutine call to the relevant Multi-function Interrupt vector locations, will take place. When the TM interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts. However, only the related MFnF flag will be automatically cleared. As the TM interrupt request flags will not be automatically cleared, they have to be cleared by the application program.

Time Base Interrupts

The function of the Time Base Interrupts is to provide regular time signal in the form of an internal interrupt. They are controlled by the overflow signals from their respective internal timer functions. When these happen their respective interrupt request flags, TB0F or TB0F, will be set. To allow the program to branch to their respective interrupt vector addresses, the global interrupt enable bit, EMI, and Time Base enable bits, TB0E or TB1E, must first be set. When the interrupt is enabled, the stack is not full and the Time Base overflows, a subroutine call to their respective interrupt vector locations will take place. When the interrupt is serviced, the TB0F or TB0F flags will be automatically reset and the EMI bit will automatically be cleared to disable other interrupts.

The purpose of the Time Base Interrupts is to provide an interrupt signal at fixed time periods. Their respective clock source, f_{PSC0} or f_{PSC1}, originates from the internal clock source f_{SYS}, $f_{SYS}/4$ or f_{SUB} and then passes through a divider, the division ratio of which is selected by programming the appropriate bits in the TB0C and TB1C registers to obtain longer interrupt periods whose value ranges. The clock source which in turn controls the Time Base interrupt period is selected using the CLKSEL0[1:0] and CLKSEL1[1:0] bits in the PSC0R and PSC1R register respectively.



Time Base Interrupts

- **PSC0R Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	CLKSEL01	CLKSEL00
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **CLKSEL01~CLKSEL00:** Prescaler 0 clock source f_{PSC0} selection
00: f_{SYS}
01: $f_{SYS}/4$
1x: f_{SUB}

- **PSC1R Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	CLKSEL11	CLKSEL10
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **CLKSEL11~CLKSEL10:** Prescaler 1 clock source f_{PSC1} selection
00: f_{SYS}
01: $f_{SYS}/4$
1x: f_{SUB}

- **TB0C Register**

Bit	7	6	5	4	3	2	1	0
Name	TB0ON	—	—	—	—	TB02	TB01	TB00
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	0	0	0

Bit 7 **TB0ON:** Time Base 0 enable control
0: Disable
1: Enable

Bit 6~3 Unimplemented, read as "0"

Bit 2~0 **TB02~TB00:** Time Base 0 time-out period selection
000: $2^8/f_{PSC0}$
001: $2^9/f_{PSC0}$
010: $2^{10}/f_{PSC0}$
011: $2^{11}/f_{PSC0}$
100: $2^{12}/f_{PSC0}$
101: $2^{13}/f_{PSC0}$
110: $2^{14}/f_{PSC0}$
111: $2^{15}/f_{PSC0}$

• TB1C Register

Bit	7	6	5	4	3	2	1	0
Name	TB1ON	—	—	—	—	TB12	TB11	TB10
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	0	0	0

Bit 7 **TB1ON:** Time Base 1 enable control
0: Disable
1: Enable

Bit 6~3 Unimplemented, read as "0"

Bit 2~0 **TB12~TB10:** Time Base 1 time-out period selection
000: $2^8/f_{PSC1}$
001: $2^9/f_{PSC1}$
010: $2^{10}/f_{PSC1}$
011: $2^{11}/f_{PSC1}$
100: $2^{12}/f_{PSC1}$
101: $2^{13}/f_{PSC1}$
110: $2^{14}/f_{PSC1}$
111: $2^{15}/f_{PSC1}$

Serial Interface Module Interrupt

The Serial Interface Module Interrupt is also known as the SIM interrupt. A SIM Interrupt request will take place when the SIM Interrupt request flag, SIMF, is set, which occurs when a byte of data has been received or transmitted by the SIM interface, an I^2C slave address match or I^2C bus time-out occurs. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and SIM Interrupt enable bit, SIME, must first be set. When the interrupt is enabled, the stack is not full and any of the above described situations occurs, a subroutine call to the corresponding SIM Interrupt vector, will take place. When the interrupt is serviced, the SIMF flag will be automatically cleared and the EMI bit will be automatically cleared to disable other interrupts.

EEPROM Interrupt

An EEPROM Interrupt request will take place when the EEPROM Interrupt request flag, DEF, is set, which occurs when an EEPROM erase or write cycle ends. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and EEPROM Interrupt enable bit, DEE, must first be set. When the interrupt is enabled, the stack is not full and an EEPROM erase or write cycle ends, a subroutine call to the corresponding EEPROM Interrupt vector will take place. When the interrupt is serviced, the DEF flag will be automatically cleared and the EMI bit will be automatically cleared to disable other interrupts.

LVD Interrupt

An LVD Interrupt request will take place when the LVD Interrupt request flag, LVF, is set, which occurs when the Low Voltage Detector function detects a low power supply voltage. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and LVD Interrupt enable bit, LVE, must first be set. When the interrupt is enabled, the stack is not full and a low voltage condition occurs, a subroutine call to the corresponding LVD Interrupt vector, will take place. When the interrupt is serviced, the LVF flag will be automatically cleared and the EMI bit will be automatically cleared to disable other interrupts.

Interrupt Wake-up Function

Each of the interrupt functions has the capability of waking up the microcontroller when in the SLEEP or IDLE Mode. A wake-up is generated when an interrupt request flag changes from low to high and is independent of whether the interrupt is enabled or not. Therefore, even though the device is in the SLEEP or IDLE Mode and its system oscillator stopped, situations such as external edge transitions on the external interrupt pins or a low power supply voltage may cause their respective interrupt flag to be set high and consequently generate an interrupt. Care must therefore be taken if spurious wake-up situations are to be avoided. If an interrupt wake-up function is to be disabled then the corresponding interrupt request flag should be set high before the device enters the SLEEP or IDLE Mode. The interrupt enable bits have no effect on the interrupt wake-up function.

Programming Considerations

By disabling the relevant interrupt enable bits, a requested interrupt can be prevented from being serviced, however, once an interrupt request flag is set, it will remain in this condition in the interrupt register until the corresponding interrupt is serviced or until the request flag is cleared by the application program.

Where a certain interrupt is contained within a Multi-function interrupt, then when the interrupt service routine is executed, as only the Multi-function interrupt request flags, MFnF, will be automatically cleared, the individual request flag for the function needs to be cleared by the application program.

It is recommended that programs do not use the “CALL” instruction within the interrupt service subroutine. Interrupts often occur in an unpredictable manner or need to be serviced immediately. If only one stack is left and the interrupt is not well controlled, the original control sequence will be damaged once a CALL subroutine is executed in the interrupt subroutine.

Every interrupt has the capability of waking up the microcontroller when it is in the SLEEP or IDLE Mode, the wake up being generated when the interrupt request flag changes from low to high. If it is required to prevent a certain interrupt from waking up the microcontroller then its respective request flag should be first set high before enter SLEEP or IDLE Mode.

As only the Program Counter is pushed onto the stack, then when the interrupt is serviced, if the contents of the accumulator, status register or other registers are altered by the interrupt service program, their contents should be saved to the memory at the beginning of the interrupt service routine.

To return from an interrupt subroutine, either a RET or RETI instruction may be executed. The RETI instruction in addition to executing a return to the main program also automatically sets the EMI bit high to allow further interrupts. The RET instruction however only executes a return to the main program leaving the EMI bit in its present zero state and therefore disabling the execution of further interrupts.

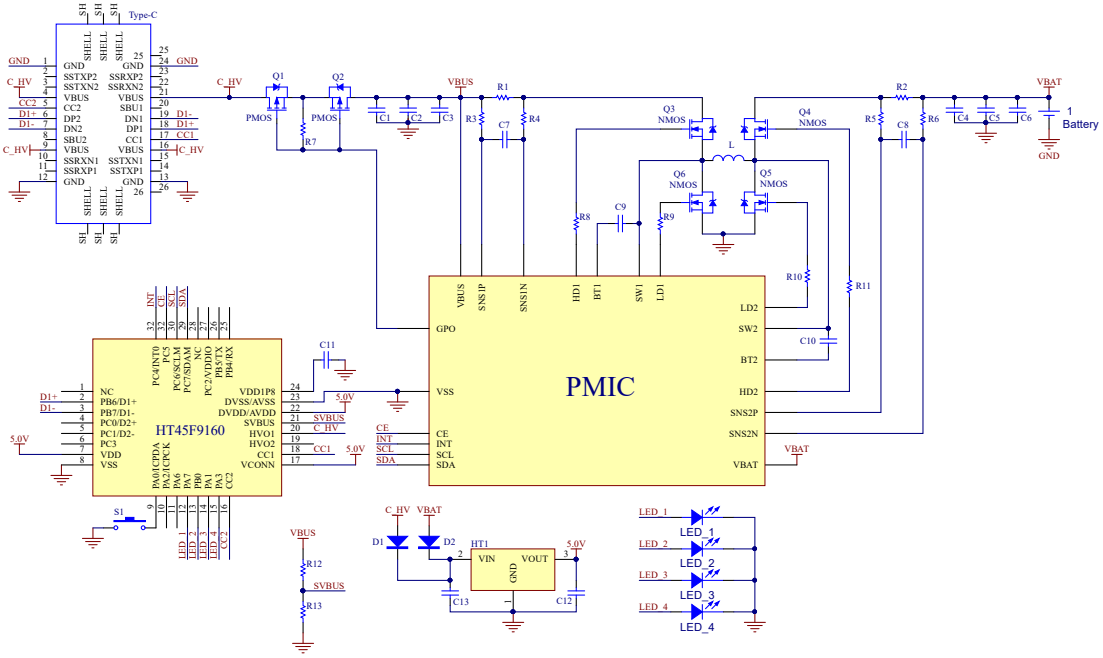
Configuration Options

Configuration options refer to certain options within the MCU that are programmed into the device during the programming process. During the development process, these options are selected using the HT-IDE software development tools. All options must be defined for proper system function, the details of which are shown in the table.

No.	Options
Oscillator Option	
1	HIRC Frequency Selection – f_{HIRC}: 12MHz, 16MHz or 20MHz

Note: When the HIRC has been configured at a frequency shown in this table, the HIRC1 and HIRC0 bits should also be setup to select the same frequency to achieve the HIRC frequency accuracy specified in the A.C. Characteristics.

Application Circuits



Instruction Set

Introduction

Central to the successful operation of any microcontroller is its instruction set, which is a set of program instruction codes that directs the microcontroller to perform certain operations. In the case of Holtek microcontroller, a comprehensive and flexible set of over 60 instructions is provided to enable programmers to implement their application with the minimum of programming overheads.

For easier understanding of the various instruction codes, they have been subdivided into several functional groupings.

Instruction Timing

Most instructions are implemented within one instruction cycle. The exceptions to this are branch, call, or table read instructions where two instruction cycles are required. One instruction cycle is equal to 4 system clock cycles, therefore in the case of an 8MHz system oscillator, most instructions would be implemented within 0.5μs and branch or call instructions would be implemented within 1μs. Although instructions which require one more cycle to implement are generally limited to the JMP, CALL, RET, RETI and table read instructions, it is important to realize that any other instructions which involve manipulation of the Program Counter Low register or PCL will also take one more cycle to implement. As instructions which change the contents of the PCL will imply a direct jump to that new address, one more cycle will be required. Examples of such instructions would be "CLR PCL" or "MOV PCL, A". For the case of skip instructions, it must be noted that if the result of the comparison involves a skip operation then this will also take one more cycle, if no skip is involved then only one cycle is required.

Moving and Transferring Data

The transfer of data within the microcontroller program is one of the most frequently used operations. Making use of several kinds of MOV instructions, data can be transferred from registers to the Accumulator and vice-versa as well as being able to move specific immediate data directly into the Accumulator. One of the most important data transfer applications is to receive data from the input ports and transfer data to the output ports.

Arithmetic Operations

The ability to perform certain arithmetic operations and data manipulation is a necessary feature of most microcontroller applications. Within the Holtek microcontroller instruction set are a range of add and subtract instruction mnemonics to enable the necessary arithmetic to be carried out. Care must be taken to ensure correct handling of carry and borrow data when results exceed 255 for addition and less than 0 for subtraction. The increment and decrement instructions such as INC, INCA, DEC and DECA provide a simple means of increasing or decreasing by a value of one of the values in the destination specified.

Logical and Rotate Operation

The standard logical operations such as AND, OR, XOR and CPL all have their own instruction within the Holtek microcontroller instruction set. As with the case of most instructions involving data manipulation, data must pass through the Accumulator which may involve additional programming steps. In all logical data operations, the zero flag may be set if the result of the operation is zero. Another form of logical data manipulation comes from the rotate instructions such as RR, RL, RRC and RLC which provide a simple means of rotating one bit right or left. Different rotate instructions exist depending on program requirements. Rotate instructions are useful for serial port programming applications where data can be rotated from an internal register into the Carry bit from where it can be examined and the necessary serial bit set high or low. Another application which rotate data operations are used is to implement multiplication and division calculations.

Branches and Control Transfer

Program branching takes the form of either jumps to specified locations using the JMP instruction or to a subroutine using the CALL instruction. They differ in the sense that in the case of a subroutine call, the program must return to the instruction immediately when the subroutine has been carried out. This is done by placing a return instruction "RET" in the subroutine which will cause the program to jump back to the address right after the CALL instruction. In the case of a JMP instruction, the program simply jumps to the desired location. There is no requirement to jump back to the original jumping off point as in the case of the CALL instruction. One special and extremely useful set of branch instructions are the conditional branches. Here a decision is first made regarding the condition of a certain data memory or individual bits. Depending upon the conditions, the program will continue with the next instruction or skip over it and jump to the following instruction. These instructions are the key to decision making and branching within the program perhaps determined by the condition of certain input switches or by the condition of internal data bits.

Bit Operations

The ability to provide single bit operations on Data Memory is an extremely flexible feature of all Holtek microcontrollers. This feature is especially useful for output port bit programming where individual bits or port pins can be directly set high or low using either the "SET [m].i" or "CLR [m].i" instructions respectively. The feature removes the need for programmers to first read the 8-bit output port, manipulate the input data to ensure that other bits are not changed and then output the port with the correct new data. This read-modify-write process is taken care of automatically when these bit operation instructions are used.

Table Read Operations

Data storage is normally implemented by using registers. However, when working with large amounts of fixed data, the volume involved often makes it inconvenient to store the fixed data in the Data Memory. To overcome this problem, Holtek microcontrollers allow an area of Program Memory to be setup as a table where data can be directly stored. A set of easy to use instructions provides the means by which this fixed data can be referenced and retrieved from the Program Memory.

Other Operations

In addition to the above functional instructions, a range of other instructions also exist such as the "HALT" instruction for Power-down operations and instructions to control the operation of the Watchdog Timer for reliable program operations under extreme electric or electromagnetic environments. For their relevant operations, refer to the functional related sections.

Instruction Set Summary

The instructions related to the data memory access in the following table can be used when the desired data memory is located in Data Memory sector 0.

Table Conventions

x: Bits immediate data
m: Data Memory address
A: Accumulator
i: 0~7 number of bits
addr: Program memory address

Mnemonic	Description	Cycles	Flag Affected
Arithmetic			
ADD A,[m]	Add Data Memory to ACC	1	Z, C, AC, OV, SC
ADDM A,[m]	Add ACC to Data Memory	1Note	Z, C, AC, OV, SC
ADD A,x	Add immediate data to ACC	1	Z, C, AC, OV, SC
ADC A,[m]	Add Data Memory to ACC with Carry	1	Z, C, AC, OV, SC
ADCM A,[m]	Add ACC to Data memory with Carry	1Note	Z, C, AC, OV, SC
SUB A,x	Subtract immediate data from the ACC	1	Z, C, AC, OV, SC, CZ
SUB A,[m]	Subtract Data Memory from ACC	1	Z, C, AC, OV, SC, CZ
SUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory	1Note	Z, C, AC, OV, SC, CZ
SBC A,x	Subtract immediate data from ACC with Carry	1	Z, C, AC, OV, SC, CZ
SBC A,[m]	Subtract Data Memory from ACC with Carry	1	Z, C, AC, OV, SC, CZ
SBCM A,[m]	Subtract Data Memory from ACC with Carry, result in Data Memory	1Note	Z, C, AC, OV, SC, CZ
DAA [m]	Decimal adjust ACC for Addition with result in Data Memory	1Note	C
Logic Operation			
AND A,[m]	Logical AND Data Memory to ACC	1	Z
OR A,[m]	Logical OR Data Memory to ACC	1	Z
XOR A,[m]	Logical XOR Data Memory to ACC	1	Z
ANDM A,[m]	Logical AND ACC to Data Memory	1Note	Z
ORM A,[m]	Logical OR ACC to Data Memory	1Note	Z
XORM A,[m]	Logical XOR ACC to Data Memory	1Note	Z
AND A,x	Logical AND immediate Data to ACC	1	Z
OR A,x	Logical OR immediate Data to ACC	1	Z
XOR A,x	Logical XOR immediate Data to ACC	1	Z
CPL [m]	Complement Data Memory	1Note	Z
CPLA [m]	Complement Data Memory with result in ACC	1	Z
Increment & Decrement			
INCA [m]	Increment Data Memory with result in ACC	1	Z
INC [m]	Increment Data Memory	1Note	Z
DECA [m]	Decrement Data Memory with result in ACC	1	Z
DEC [m]	Decrement Data Memory	1Note	Z
Rotate			
RRA [m]	Rotate Data Memory right with result in ACC	1	None
RR [m]	Rotate Data Memory right	1Note	None
RRCA [m]	Rotate Data Memory right through Carry with result in ACC	1	C
RRC [m]	Rotate Data Memory right through Carry	1Note	C
RLA [m]	Rotate Data Memory left with result in ACC	1	None
RL [m]	Rotate Data Memory left	1Note	None
RLCA [m]	Rotate Data Memory left through Carry with result in ACC	1	C
RLC [m]	Rotate Data Memory left through Carry	1Note	C

Mnemonic	Description	Cycles	Flag Affected
Data Move			
MOV A,[m]	Move Data Memory to ACC	1	None
MOV [m],A	Move ACC to Data Memory	1Note	None
MOV A,x	Move immediate data to ACC	1	None
Bit Operation			
CLR [m].i	Clear bit of Data Memory	1Note	None
SET [m].i	Set bit of Data Memory	1Note	None
Branch Operation			
JMP addr	Jump unconditionally	2	None
SZ [m]	Skip if Data Memory is zero	1Note	None
SZA [m]	Skip if Data Memory is zero with data movement to ACC	1Note	None
SZ [m].i	Skip if bit i of Data Memory is zero	1Note	None
SNZ [m]	Skip if Data Memory is not zero	1Note	None
SNZ [m].i	Skip if bit i of Data Memory is not zero	1Note	None
SIZ [m]	Skip if increment Data Memory is zero	1Note	None
SDZ [m]	Skip if decrement Data Memory is zero	1Note	None
SIZA [m]	Skip if increment Data Memory is zero with result in ACC	1Note	None
SDZA [m]	Skip if decrement Data Memory is zero with result in ACC	1Note	None
CALL addr	Subroutine call	2	None
RET	Return from subroutine	2	None
RET A,x	Return from subroutine and load immediate data to ACC	2	None
RETI	Return from interrupt	2	None
Table Read Operation			
TABRD [m]	Read table (specific page) to TBLH and Data Memory	2Note	None
TABRDL [m]	Read table (last page) to TBLH and Data Memory	2Note	None
ITABRD [m]	Increment table pointer TBLP first and Read table (specific page) to TBLH and Data Memory	2Note	None
ITABRDL [m]	Increment table pointer TBLP first and Read table (last page) to TBLH and Data Memory	2Note	None
Miscellaneous			
NOP	No operation	1	None
CLR [m]	Clear Data Memory	1Note	None
SET [m]	Set Data Memory	1Note	None
CLR WDT	Clear Watchdog Timer	1	TO, PDF
SWAP [m]	Swap nibbles of Data Memory	1Note	None
SWAPA [m]	Swap nibbles of Data Memory with result in ACC	1	None
HALT	Enter power down mode	1	TO, PDF

Note: 1. For skip instructions, if the result of the comparison involves a skip then two cycles are required, if no skip takes place only one cycle is required.

2. Any instruction which changes the contents of the PCL will also require 2 cycles for execution.

Extended Instruction Set

The extended instructions are used to support the full range address access for the data memory. When the accessed data memory is located in any data memory sector except sector 0, the extended instruction can be used to directly access the data memory instead of using the indirect addressing access. This can not only reduce the use of Flash memory space but also improve the CPU execution efficiency.

Mnemonic	Description	Cycles	Flag Affected
Arithmetic			
LADD A,[m]	Add Data Memory to ACC	2	Z, C, AC, OV, SC
LADDM A,[m]	Add ACC to Data Memory	2Note	Z, C, AC, OV, SC
LADC A,[m]	Add Data Memory to ACC with Carry	2	Z, C, AC, OV, SC
LADCM A,[m]	Add ACC to Data memory with Carry	2Note	Z, C, AC, OV, SC
LSUB A,[m]	Subtract Data Memory from ACC	2	Z, C, AC, OV, SC, CZ
LSUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory	2Note	Z, C, AC, OV, SC, CZ
LSBC A,[m]	Subtract Data Memory from ACC with Carry	2	Z, C, AC, OV, SC, CZ
LSBCM A,[m]	Subtract Data Memory from ACC with Carry, result in Data Memory	2Note	Z, C, AC, OV, SC, CZ
LDAA [m]	Decimal adjust ACC for Addition with result in Data Memory	2Note	C
Logic Operation			
LAND A,[m]	Logical AND Data Memory to ACC	2	Z
LOR A,[m]	Logical OR Data Memory to ACC	2	Z
LXOR A,[m]	Logical XOR Data Memory to ACC	2	Z
LANDM A,[m]	Logical AND ACC to Data Memory	2Note	Z
LORM A,[m]	Logical OR ACC to Data Memory	2Note	Z
LXORM A,[m]	Logical XOR ACC to Data Memory	2Note	Z
LCPL [m]	Complement Data Memory	2Note	Z
LCPLA [m]	Complement Data Memory with result in ACC	2	Z
Increment & Decrement			
LINCA [m]	Increment Data Memory with result in ACC	2	Z
LINC [m]	Increment Data Memory	2Note	Z
LDECA [m]	Decrement Data Memory with result in ACC	2	Z
LDEC [m]	Decrement Data Memory	2Note	Z
Rotate			
LRRA [m]	Rotate Data Memory right with result in ACC	2	None
LRR [m]	Rotate Data Memory right	2Note	None
LRRCA [m]	Rotate Data Memory right through Carry with result in ACC	2	C
LRRC [m]	Rotate Data Memory right through Carry	2Note	C
LRLA [m]	Rotate Data Memory left with result in ACC	2	None
LRL [m]	Rotate Data Memory left	2Note	None
LRLCA [m]	Rotate Data Memory left through Carry with result in ACC	2	C
LRLC [m]	Rotate Data Memory left through Carry	2Note	C
Data Move			
LMOV A,[m]	Move Data Memory to ACC	2	None
LMOV [m],A	Move ACC to Data Memory	2Note	None
Bit Operation			
LCLR [m].i	Clear bit of Data Memory	2Note	None
LSET [m].i	Set bit of Data Memory	2Note	None

Mnemonic	Description	Cycles	Flag Affected
Branch			
LSZ [m]	Skip if Data Memory is zero	2Note	None
LSZA [m]	Skip if Data Memory is zero with data movement to ACC	2Note	None
LSNZ [m]	Skip if Data Memory is not zero	2Note	None
LSZ [m].i	Skip if bit i of Data Memory is zero	2Note	None
LSNZ [m].i	Skip if bit i of Data Memory is not zero	2Note	None
LSIZ [m]	Skip if increment Data Memory is zero	2Note	None
LSDZ [m]	Skip if decrement Data Memory is zero	2Note	None
LSIZA [m]	Skip if increment Data Memory is zero with result in ACC	2Note	None
LSDZA [m]	Skip if decrement Data Memory is zero with result in ACC	2Note	None
Table Read			
LTABRD [m]	Read table (specific page) to TBLH and Data Memory	3Note	None
LTABRDL [m]	Read table (last page) to TBLH and Data Memory	3Note	None
LITABRD [m]	Increment table pointer TBLP first and Read table (specific page) to TBLH and Data Memory	3Note	None
LITABRDL [m]	Increment table pointer TBLP first and Read table (last page) to TBLH and Data Memory	3Note	None
Miscellaneous			
LCLR [m]	Clear Data Memory	2Note	None
LSET [m]	Set Data Memory	2Note	None
LSWAP [m]	Swap nibbles of Data Memory	2Note	None
LSWAPA [m]	Swap nibbles of Data Memory with result in ACC	2	None

Note: 1. For these extended skip instructions, if the result of the comparison involves a skip then three cycles are required, if no skip takes place two cycles is required.

2. Any extended instruction which changes the contents of the PCL register will also require three cycles for execution.

Instruction Definition

ADC A,[m] Add Data Memory to ACC with Carry

Description The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the Accumulator.

Operation ACC ← ACC + [m] + C

Affected flag(s) OV, Z, AC, C, SC

ADCM A,[m] Add ACC to Data Memory with Carry

Description The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the specified Data Memory.

Operation [m] ← ACC + [m] + C

Affected flag(s) OV, Z, AC, C, SC

ADD A,[m] Add Data Memory to ACC

Description The contents of the specified Data Memory and the Accumulator are added. The result is stored in the Accumulator.

Operation ACC ← ACC + [m]

Affected flag(s) OV, Z, AC, C, SC

ADD A,x Add immediate data to ACC

Description The contents of the Accumulator and the specified immediate data are added. The result is stored in the Accumulator.

Operation ACC ← ACC + x

Affected flag(s) OV, Z, AC, C, SC

ADDM A,[m] Add ACC to Data Memory

Description The contents of the specified Data Memory and the Accumulator are added. The result is stored in the specified Data Memory.

Operation [m] ← ACC + [m]

Affected flag(s) OV, Z, AC, C, SC

AND A,[m] Logical AND Data Memory to ACC

Description Data in the Accumulator and the specified Data Memory perform a bitwise logical AND operation. The result is stored in the Accumulator.

Operation ACC ← ACC "AND" [m]

Affected flag(s) Z

AND A,x Logical AND immediate data to ACC

Description Data in the Accumulator and the specified immediate data perform a bit wise logical AND operation. The result is stored in the Accumulator.

Operation ACC ← ACC "AND" x

Affected flag(s) Z

ANDM A,[m] Logical AND ACC to Data Memory

Description Data in the specified Data Memory and the Accumulator perform a bitwise logical AND operation. The result is stored in the Data Memory.

Operation [m] ← ACC "AND" [m]

Affected flag(s) Z

CALL addr Subroutine call

Description Unconditionally calls a subroutine at the specified address. The Program Counter then increments by 1 to obtain the address of the next instruction which is then pushed onto the stack. The specified address is then loaded and the program continues execution from this new address. As this instruction requires an additional operation, it is a two cycle instruction.

Operation Stack ← Program Counter + 1
Program Counter ← addr

Affected flag(s) None

CLR [m] Clear Data Memory

Description Each bit of the specified Data Memory is cleared to 0.

Operation [m] ← 00H

Affected flag(s) None

CLR [m].i Clear bit of Data Memory

Description Bit i of the specified Data Memory is cleared to 0.

Operation [m].i ← 0

Affected flag(s) None

CLR WDT Clear Watchdog Timer

Description The TO, PDF flags and the WDT are all cleared.

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CPL [m] Complement Data Memory

Description Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa.

Operation [m] ← [m]

Affected flag(s) Z

CPLA [m] Complement Data Memory with result in ACC

Description Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa. The complemented result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC ← [m]

Affected flag(s) Z

DAA [m] Decimal-Adjust ACC for addition with result in Data Memory

Description Convert the contents of the Accumulator value to a BCD (Binary Coded Decimal) value resulting from the previous addition of two BCD variables. If the low nibble is greater than 9 or if AC flag is set, then a value of 6 will be added to the low nibble. Otherwise the low nibble remains unchanged. If the high nibble is greater than 9 or if the C flag is set, then a value of 6 will be added to the high nibble. Essentially, the decimal conversion is performed by adding 00H, 06H, 60H or 66H depending on the Accumulator and flag conditions. Only the C flag may be affected by this instruction which indicates that if the original BCD sum is greater than 100, it allows multiple precision decimal addition.

Operation [m] ← ACC + 00H or
[m] ← ACC + 06H or
[m] ← ACC + 60H or
[m] ← ACC + 66H

Affected flag(s) C

DEC [m] Decrement Data Memory

Description Data in the specified Data Memory is decremented by 1.

Operation [m] ← [m] − 1

Affected flag(s) Z

DECA [m] Decrement Data Memory with result in ACC

Description Data in the specified Data Memory is decremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation ACC ← [m] − 1

Affected flag(s) Z

HALT Enter power down mode

Description This instruction stops the program execution and turns off the system clock. The contents of the Data Memory and registers are retained. The WDT and prescaler are cleared. The power down flag PDF is set and the WDT time-out flag TO is cleared.

Operation TO ← 0
PDF ← 1

Affected flag(s) TO, PDF

INC [m] Increment Data Memory

Description Data in the specified Data Memory is incremented by 1.

Operation [m] ← [m] + 1

Affected flag(s) Z

INCA [m] Increment Data Memory with result in ACC

Description Data in the specified Data Memory is incremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation ACC ← [m] + 1

Affected flag(s) Z

JMP addr Jump unconditionally

Description The contents of the Program Counter are replaced with the specified address. Program execution then continues from this new address. As this requires the insertion of a dummy instruction while the new address is loaded, it is a two cycle instruction.

Operation Program Counter ← addr

Affected flag(s) None

MOV A,[m] Move Data Memory to ACC

Description The contents of the specified Data Memory are copied to the Accumulator.

Operation ACC ← [m]

Affected flag(s) None

MOV A,x Move immediate data to ACC

Description The immediate data specified is loaded into the Accumulator.

Operation ACC ← x

Affected flag(s) None

MOV [m],A Move ACC to Data Memory

Description The contents of the Accumulator are copied to the specified Data Memory.

Operation [m] ← ACC

Affected flag(s) None

NOP No operation

Description No operation is performed. Execution continues with the next instruction.

Operation No operation

Affected flag(s) None

OR A,[m] Logical OR Data Memory to ACC

Description Data in the Accumulator and the specified Data Memory perform a bitwise logical OR operation. The result is stored in the Accumulator.

Operation ACC ← ACC "OR" [m]

Affected flag(s) Z

OR A,x Logical OR immediate data to ACC

Description Data in the Accumulator and the specified immediate data perform a bitwise logical OR operation. The result is stored in the Accumulator.

Operation ACC ← ACC "OR" x

Affected flag(s) Z

ORM A,[m] Logical OR ACC to Data Memory

Description Data in the specified Data Memory and the Accumulator perform a bitwise logical OR operation. The result is stored in the Data Memory.

Operation [m] ← ACC "OR" [m]

Affected flag(s) Z

RET Return from subroutine

Description The Program Counter is restored from the stack. Program execution continues at the restored address.

Operation Program Counter ← Stack

Affected flag(s) None

RET A,x Return from subroutine and load immediate data to ACC

Description The Program Counter is restored from the stack and the Accumulator loaded with the specified immediate data. Program execution continues at the restored address.

Operation Program Counter ← Stack
ACC ← x

Affected flag(s) None

RETI Return from interrupt

Description The Program Counter is restored from the stack and the interrupts are re-enabled by setting the EMI bit. EMI is the master interrupt global enable bit. If an interrupt was pending when the RETI instruction is executed, the pending Interrupt routine will be processed before returning to the main program.

Operation Program Counter ← Stack
EMI ← 1

Affected flag(s) None

RL [m] Rotate Data Memory left

Description The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0.

Operation [m].(i+1) ← [m].i; (i=0~6)
[m].0 ← [m].7

Affected flag(s) None

RLA [m] Rotate Data Memory left with result in ACC

Description The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC.(i+1) ← [m].i; (i=0~6)
ACC.0 ← [m].7

Affected flag(s) None

RLC [m] Rotate Data Memory left through Carry

Description The contents of the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into bit 0.

Operation [m].(i+1) ← [m].i; (i=0~6)
[m].0 ← C
C ← [m].7

Affected flag(s) C

RLCA [m] Rotate Data Memory left through Carry with result in ACC

Description Data in the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into the bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC.(i+1) ← [m].i; (i=0~6)
ACC.0 ← C
C ← [m].7

Affected flag(s) C

RR [m] Rotate Data Memory right

Description The contents of the specified Data Memory are rotated right by 1 bit with bit 0 rotated into bit 7.

Operation [m].i ← [m].(i+1); (i=0~6)
[m].7 ← [m].0

Affected flag(s) None

RRA [m] Rotate Data Memory right with result in ACC

Description Data in the specified Data Memory is rotated right by 1 bit with bit 0 rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC.i ← [m].(i+1); (i=0~6)
ACC.7 ← [m].0

Affected flag(s) None

RRC [m] Rotate Data Memory right through Carry

Description The contents of the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7.

Operation [m].i ← [m].(i+1); (i=0~6)
[m].7 ← C
C ← [m].0

Affected flag(s) C

RRCA [m] Rotate Data Memory right through Carry with result in ACC

Description Data in the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC.i ← [m].(i+1); (i=0~6)
ACC.7 ← C
C ← [m].0

Affected flag(s) C

SBC A,[m] Subtract Data Memory from ACC with Carry

Description The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation ACC ← ACC − [m] − C

Affected flag(s) OV, Z, AC, C, SC, CZ

SBC A, x Subtract immediate data from ACC with Carry

Description The immediate data and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation ACC ← ACC – [m] – C

Affected flag(s) OV, Z, AC, C, SC, CZ

SBCM A,[m] Subtract Data Memory from ACC with Carry and result in Data Memory

Description The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation [m] ← ACC − [m] − C

Affected flag(s) OV, Z, AC, C, SC, CZ

SDZ [m] Skip if decrement Data Memory is 0

Description The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation [m] ← [m] − 1
Skip if [m]=0

Affected flag(s) None

SDZA [m] Skip if decrement Data Memory is zero with result in ACC

Description The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation ACC ← [m] − 1
Skip if ACC=0

Affected flag(s) None

SET [m] Set Data Memory

Description Each bit of the specified Data Memory is set to 1.

Operation [m] ← FFH

Affected flag(s) None

SET [m].i Set bit of Data Memory

Description Bit i of the specified Data Memory is set to 1.

Operation [m].i ← 1

Affected flag(s) None

SIZ [m] Skip if increment Data Memory is 0

Description The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation [m] ← [m] + 1
Skip if [m]=0

Affected flag(s) None

SIZA [m] Skip if increment Data Memory is zero with result in ACC

Description The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation ACC ← [m] + 1
Skip if ACC=0

Affected flag(s) None

SNZ [m].i Skip if bit i of Data Memory is not 0

Description If bit i of the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.

Operation Skip if [m].i ≠ 0

Affected flag(s) None

SNZ [m] Skip if Data Memory is not 0

Description The contents of the specified Data Memory are read out and then written back to the specified Data Memory again. If the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.

Operation Skip if [m]≠ 0

Affected flag(s) None

SUB A,[m] Subtract Data Memory from ACC

Description The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation ACC ← ACC − [m]

Affected flag(s) OV, Z, AC, C, SC, CZ

SUBM A,[m] Subtract Data Memory from ACC with result in Data Memory

Description The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation [m] ← ACC − [m]

Affected flag(s) OV, Z, AC, C, SC, CZ

SUB A,x Subtract immediate data from ACC

Description The immediate data specified by the code is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation ACC ← ACC − x

Affected flag(s) OV, Z, AC, C, SC, CZ

SWAP [m] Swap nibbles of Data Memory

Description The low-order and high-order nibbles of the specified Data Memory are interchanged.

Operation [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s) None

SWAPA [m] Swap nibbles of Data Memory with result in ACC

Description The low-order and high-order nibbles of the specified Data Memory are interchanged. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation ACC.3~ACC.0 ← [m].7~[m].4
ACC.7~ACC.4 ← [m].3~[m].0

Affected flag(s) None

SZ [m] Skip if Data Memory is 0

Description The contents of the specified Data Memory are read out and then written back to the specified Data Memory again. If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation Skip if [m]=0

Affected flag(s) None

SZA [m] Skip if Data Memory is 0 with data movement to ACC

Description The contents of the specified Data Memory are copied to the Accumulator. If the value is zero, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation ACC ← [m]
Skip if [m]=0

Affected flag(s) None

SZ [m].i Skip if bit i of Data Memory is 0

Description If bit i of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation Skip if [m].i=0

Affected flag(s) None

TABRD [m] Read table (specific page) to TBLH and Data Memory

Description The low byte of the program code (specific page) addressed by the table pointer (TBLP and TBHP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

TABRDL [m] Read table (last page) to TBLH and Data Memory

Description The low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

ITABRD [m] Increment table pointer low byte first and read table (specific page) to TBLH and Data Memory Description Increment table pointer low byte, TBLP, first and then the program code (specific page) addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

ITABRDL [m] Increment table pointer low byte first and read table (last page) to TBLH and Data Memory

Description Increment table pointer low byte, TBLP, first and then the low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

XOR A,[m] Logical XOR Data Memory to ACC

Description Data in the Accumulator and the specified Data Memory perform a bitwise logical XOR operation. The result is stored in the Accumulator.

Operation ACC ← ACC "XOR" [m]

Affected flag(s) Z

XORM A,[m] Logical XOR ACC to Data Memory

Description Data in the specified Data Memory and the Accumulator perform a bitwise logical XOR operation. The result is stored in the Data Memory.

Operation [m] ← ACC "XOR" [m]

Affected flag(s) Z

XOR A,x Logical XOR immediate data to ACC

Description Data in the Accumulator and the specified immediate data perform a bitwise logical XOR operation. The result is stored in the Accumulator.

Operation ACC ← ACC "XOR" x

Affected flag(s) Z

Extended Instruction Definition

The extended instructions are used to directly access the data stored in any data memory sections.

LADC A,[m] Add Data Memory to ACC with Carry

Description The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the Accumulator.

Operation ACC ← ACC + [m] + C

Affected flag(s) OV, Z, AC, C, SC

LADCM A,[m] Add ACC to Data Memory with Carry

Description The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the specified Data Memory.

Operation [m] ← ACC + [m] + C

Affected flag(s) OV, Z, AC, C, SC

LADD A,[m] Add Data Memory to ACC

Description The contents of the specified Data Memory and the Accumulator are added. The result is stored in the Accumulator.

Operation ACC ← ACC + [m]

Affected flag(s) OV, Z, AC, C, SC

LADDM A,[m] Add ACC to Data Memory

Description The contents of the specified Data Memory and the Accumulator are added. The result is stored in the specified Data Memory.

Operation [m] ← ACC + [m]

Affected flag(s) OV, Z, AC, C, SC

LAND A,[m] Logical AND Data Memory to ACC

Description Data in the Accumulator and the specified Data Memory perform a bitwise logical AND operation. The result is stored in the Accumulator.

Operation ACC ← ACC "AND" [m]

Affected flag(s) Z

LANDM A,[m] Logical AND ACC to Data Memory

Description Data in the specified Data Memory and the Accumulator perform a bitwise logical AND operation. The result is stored in the Data Memory.

Operation [m] ← ACC "AND" [m]

Affected flag(s) Z

LCLR [m] Clear Data Memory

Description Each bit of the specified Data Memory is cleared to 0.

Operation [m] ← 00H

Affected flag(s) None

LCLR [m].i Clear bit of Data Memory

Description Bit i of the specified Data Memory is cleared to 0.

Operation [m].i ← 0

Affected flag(s) None

LCPL [m] Complement Data Memory

Description Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa.

Operation [m] ← [m]

Affected flag(s) Z

LCPLA [m] Complement Data Memory with result in ACC

Description Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa. The complemented result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC ← [m]

Affected flag(s) Z

LDAA [m] Decimal-Adjust ACC for addition with result in Data Memory

Description Convert the contents of the Accumulator value to a BCD (Binary Coded Decimal) value resulting from the previous addition of two BCD variables. If the low nibble is greater than 9 or if AC flag is set, then a value of 6 will be added to the low nibble. Otherwise the low nibble remains unchanged. If the high nibble is greater than 9 or if the C flag is set, then a value of 6 will be added to the high nibble. Essentially, the decimal conversion is performed by adding 00H, 06H, 60H or 66H depending on the Accumulator and flag conditions. Only the C flag may be affected by this instruction which indicates that if the original BCD sum is greater than 100, it allows multiple precision decimal addition.

Operation [m] ← ACC + 00H or
[m] ← ACC + 06H or
[m] ← ACC + 60H or
[m] ← ACC + 66H

Affected flag(s) C

LDEC [m] Decrement Data Memory

Description Data in the specified Data Memory is decremented by 1.

Operation [m] ← [m] − 1

Affected flag(s) Z

LDECA [m] Decrement Data Memory with result in ACC

Description Data in the specified Data Memory is decremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation ACC ← [m] − 1

Affected flag(s) Z

LINC [m] Increment Data Memory

Description Data in the specified Data Memory is incremented by 1.

Operation [m] ← [m] + 1

Affected flag(s) Z

LINCA [m] Increment Data Memory with result in ACC

Description Data in the specified Data Memory is incremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation ACC ← [m] + 1

Affected flag(s) Z

LMOV A,[m] Move Data Memory to ACC

Description The contents of the specified Data Memory are copied to the Accumulator.

Operation ACC ← [m]

Affected flag(s) None

LMOV [m],A Move ACC to Data Memory

Description The contents of the Accumulator are copied to the specified Data Memory.

Operation [m] ← ACC

Affected flag(s) None

LOR A,[m] Logical OR Data Memory to ACC

Description Data in the Accumulator and the specified Data Memory perform a bitwise logical OR operation. The result is stored in the Accumulator.

Operation ACC ← ACC "OR" [m]

Affected flag(s) Z

LORM A,[m] Logical OR ACC to Data Memory

Description Data in the specified Data Memory and the Accumulator perform a bitwise logical OR operation. The result is stored in the Data Memory.

Operation [m] ← ACC "OR" [m]

Affected flag(s) Z

LRL [m] Rotate Data Memory left

Description The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0.

Operation [m].(i+1) ← [m].i; (i=0~6)
[m].0 ← [m].7

Affected flag(s) None

LRLA [m] Rotate Data Memory left with result in ACC

Description The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC.(i+1) ← [m].i; (i=0~6)
ACC.0 ← [m].7

Affected flag(s) None

LRLC [m] Rotate Data Memory left through Carry

Description The contents of the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into bit 0.

Operation [m].(i+1) ← [m].i; (i=0~6)
[m].0 ← C
C ← [m].7

Affected flag(s) C

LRLCA [m] Rotate Data Memory left through Carry with result in ACC

Description Data in the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into the bit 0. The rotated result is stored in theAccumulator and the contents of the Data Memory remain unchanged.

Operation ACC.(i+1) ← [m].i; (i=0~6)
ACC.0 ← C
C ← [m].7

Affected flag(s) C

LRR [m] Rotate Data Memory right

Description The contents of the specified Data Memory are rotated right by 1 bit with bit 0 rotated into bit 7.

Operation [m].i ← [m].(i+1); (i=0~6)
[m].7 ← [m].0

Affected flag(s) None

LRRA [m] Rotate Data Memory right with result in ACC

Description Data in the specified Data Memory is rotated right by 1 bit with bit 0 rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC.i ← [m].(i+1); (i=0~6)
ACC.7 ← [m].0

Affected flag(s) None

LRRC [m] Rotate Data Memory right through Carry

Description The contents of the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7.

Operation [m].i ← [m].(i+1); (i=0~6)
[m].7 ← C
C ← [m].0

Affected flag(s) C

LRRCA [m] Rotate Data Memory right through Carry with result in ACC

Description Data in the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC.i ← [m].(i+1); (i=0~6)
ACC.7 ← C
C ← [m].0

Affected flag(s) C

LSBC A,[m] Subtract Data Memory from ACC with Carry

Description The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation ACC ← ACC − [m] − C

Affected flag(s) OV, Z, AC, C, SC, CZ

LSBCM A,[m] Subtract Data Memory from ACC with Carry and result in Data Memory

Description The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation [m] ← ACC − [m] − C

Affected flag(s) OV, Z, AC, C, SC, CZ

LSDZ [m] Skip if decrement Data Memory is 0

Description The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation [m] ← [m] − 1
Skip if [m]=0

Affected flag(s) None

LSDZA [m] Skip if decrement Data Memory is zero with result in ACC

Description The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation ACC ← [m] − 1
Skip if ACC=0

Affected flag(s) None

LSET [m] Set Data Memory

Description Each bit of the specified Data Memory is set to 1.

Operation [m] ← FFH

Affected flag(s) None

LSET [m].i Set bit of Data Memory

Description Bit i of the specified Data Memory is set to 1.

Operation [m].i ← 1

Affected flag(s) None

LSIZ [m] Skip if increment Data Memory is 0

Description The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation [m] ← [m] + 1
Skip if [m]=0

Affected flag(s) None

LSIZA [m] Skip if increment Data Memory is zero with result in ACC

Description The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation ACC ← [m] + 1
Skip if ACC=0

Affected flag(s) None

LSNZ [m].i Skip if bit i of Data Memory is not 0

Description If bit i of the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is 0 the program proceeds with the following instruction.

Operation Skip if [m].i ≠ 0

Affected flag(s) None

LSNZ [m] Skip if Data Memory is not 0

Description The contents of the specified Data Memory are read out and then written to the specified Data Memory again. If the content of the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is 0 the program proceeds with the following instruction.

Operation Skip if [m] ≠ 0

Affected flag(s) None

LSUB A,[m] Subtract Data Memory from ACC

Description The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation ACC ← ACC − [m]

Affected flag(s) OV, Z, AC, C, SC, CZ

LSUBM A,[m] Subtract Data Memory from ACC with result in Data Memory

Description The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation [m] ← ACC − [m]

Affected flag(s) OV, Z, AC, C, SC, CZ

LSWAP [m] Swap nibbles of Data Memory

Description The low-order and high-order nibbles of the specified Data Memory are interchanged.

Operation [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s) None

LSWAPA [m] Swap nibbles of Data Memory with result in ACC

Description The low-order and high-order nibbles of the specified Data Memory are interchanged. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation ACC.3~ACC.0 ← [m].7~[m].4
ACC.7~ACC.4 ← [m].3~[m].0

Affected flag(s) None

LSZ [m] Skip if Data Memory is 0

Description The contents of the specified Data Memory are read out and then written to the specified Data Memory again. If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation Skip if [m]=0

Affected flag(s) None

LSZA [m] Skip if Data Memory is 0 with data movement to ACC

Description The contents of the specified Data Memory are copied to the Accumulator. If the value is zero, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation ACC ← [m]
Skip if [m]=0

Affected flag(s) None

LSZ [m].i Skip if bit i of Data Memory is 0

Description If bit i of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation Skip if [m].i=0

Affected flag(s) None

LTABRD [m] Read table (specific page) to TBLH and Data Memory

Description The low byte of the program code (specific page) addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

LTABRDL [m] Read table (last page) to TBLH and Data Memory

Description The low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

LITABRD [m] Increment table pointer low byte first and read table (specific page) to TBLH and Data Memory

Description Increment table pointer low byte, TBLP, first and then the program code (specific page) addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

LITABRDL [m] Increment table pointer low byte first and read table (last page) to TBLH and Data Memory

Description Increment table pointer low byte, TBLP, first and then the low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

LXOR A,[m] Logical XOR Data Memory to ACC

Description Data in the Accumulator and the specified Data Memory perform a bitwise logical XOR operation. The result is stored in the Accumulator.

Operation ACC ← ACC "XOR" [m]

Affected flag(s) Z

LXORM A,[m] Logical XOR ACC to Data Memory

Description Data in the specified Data Memory and the Accumulator perform a bitwise logical XOR operation. The result is stored in the Data Memory.

Operation [m] ← ACC "XOR" [m]

Affected flag(s) Z

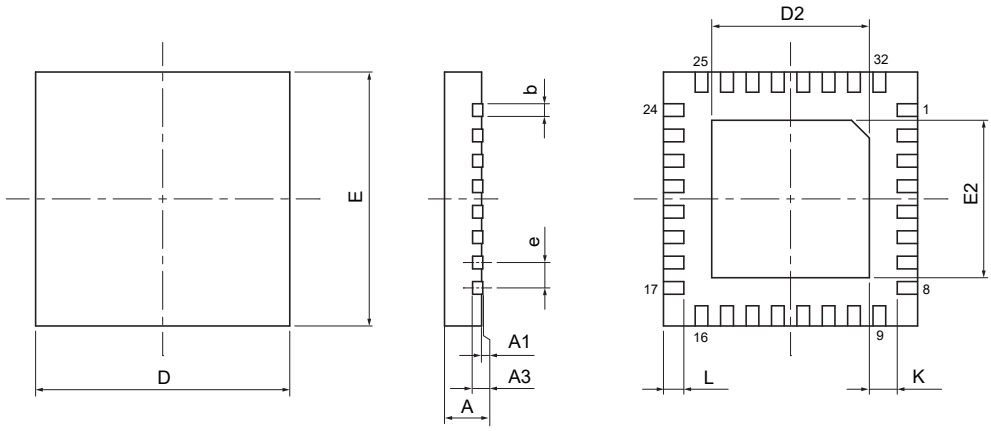
Package Information

Note that the package information provided here is for consultation purposes only. As this information may be updated at regular intervals users are reminded to consult the Holtek website for the latest version of the Package/Carton Information.

Additional supplementary information with regard to packaging is listed below. Click on the relevant section to be transferred to the relevant website page.

- Package Information (include Outline Dimensions, Product Tape and Reel Specifications)
- The Operation Instruction of Packing Materials
- Carton information

SAW Type 32-pin QFN (4mm×4mm×0.75mm) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.028	0.030	0.031
A1	0.000	0.001	0.002
A3		0.008 REF	
b	0.006	0.008	0.010
D		0.157 BSC	
E		0.157 BSC	
e		0.016 BSC	
D2	0.100	—	0.108
E2	0.100	—	0.108
L	0.010	—	0.018
K	0.008	—	—

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	0.70	0.75	0.80
A1	0.00	0.02	0.05
A3		0.203 REF	
b	0.15	0.20	0.25
D		4.00 BSC	
E		4.00 BSC	
e		0.40 BSC	
D2	2.55	—	2.75
E2	2.55	—	2.75
L	0.25	—	0.45
K	0.20	—	—